НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

«На правах рукопису»
УДК __004.056____

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Програмне забезпечення інформаційно-комунікаційних систем»
зі спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Система захисту сервера автентифікації RADIUS від DoS атак»

Виконала:
студентка VI курсу, групи ІТ-93мп
Некряч Дарія Олександрівна _____

Керівник:
доцент каф. АУТС, к.т.н.
Букасов Максим Михайлович _____

Рецензент:

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Програмне забезпечення інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенці

Некряч Дарії Олександрівні

1. Тема дисертації «Система захисту сервера автентифікації RADIUS від DoS атак», науковий керівник дисертації Букасов Максим Михайлович, к.т.н., доцент каф. АУТС затверджені наказом по університету від «26» 10 2020 р. №3132-с

2. Термін подання студенткою дисертації _____

3. Об’єкт дослідження: процес захисту сервера автентифікації RADIUS від DoS атак

4. Вихідні дані: фільтрація RADIUS-пакетів з подальшим їх блокуванням у випадку сумнівної активності

5. Перелік завдань, які потрібно розробити: розробка системи захисту сервера автентифікації RADIUS від DoS атак на основі алгоритму фільтрації пакетів

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: діаграма сценаріїв використання, структурна схема, діаграма розгортання, ER-діаграма бази даних сервера захисту, діаграма послідовності процесу фільтрації запиту на автентифікацію, блок-схема алгоритму фільтрації пакеті, діаграма комунікацій для випадку обробки RadSec пакетів, діаграма комунікацій аналізу пакетів

7. Орієнтовний перелік публікацій: Winter InfoCom Advanced Solutions 2020

9. Дата видачі завдання 03.09.2020

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Ознайомлення з завданням	03.09.2020	
2	Огляд та аналіз існуючих рішень та формування вимог	14.09.2020	
3	Вибір технологій	28.09.2020	
4	Проектування архітектури системи	05.10.2020	
5	Розроблення програмного забезпечення	12.10.2020	
6	Тестування системи	16.11.2020	
7	Оформлення документації	30.11.2020	
8	Предзахист	02.12.2020	

Студентка Дарія НЕКРЯЧ

Науковий керівник Максим БУКАСОВ

РЕФЕРАТ

Магістерська дисертація: 102 с., 28 рис., 27 табл., 9 додатків, 26 джерел.

Актуальність. Компаніям часто необхідно відстежувати які пристрої та користувачі автентифікуються та мають доступ до корпоративної мережі. Головна мета цього – попередження або активна протидія проти різних кібератак або ж проста заборона несанкціонованого доступу. Існують рішення контролю доступу до мережі, в рамках якого для проведення автентифікації та обліку користувачів використовується RADIUS-сервер.

Також є необхідність захисту даних рішень, а саме RADIUS-серверу, від різноманітних атак, у тому числі однієї з поширених – DoS атаки, тобто створення надлишкового трафіку у вигляді непотрібних чи деформованих RADIUS пакетів.

Метою є створення системи, що дозволить проводити аналіз пакетів автентифікації та обліку протоколу RADIUS для подальшого захисту від DoS атак.

Об’єктом дослідження є система, що дозволяє захищати сервер автентифікації RADIUS від DoS атак.

Предметом є фільтрація RADIUS-пакетів з подальшим їх блокуванням у випадку сумнівної активності.

Ключові слова: DoS атака, автентифікація, RADIUS, фільтрація пакетів, блокування, захист.

ABSTRACT

Master’s thesis: 102 pages, 28 figures, 27 tables, 9 applications, 26 sources.

Relevance. Companies often need to monitor what devices and users authenticate and have access into corporate network. The main reason for this is preventing or active countermeasure to various cyber-attacks or just simple denying of unauthorized access. There are solutions like network access control that use RADIUS-server to perform authentication and accounting of users.

There is also a need to protect such solution as RADIUS-server from various attacks including one of the most common ones – DoS attacks, that create excessive traffic in form of unnecessary or broken RAIUS packets.

The goal is to create system that will allow to analyze authenticating and accounting packets of RADIUS protocol for further security against DoS-attacks.

The object of the research is the system that allows to secure authenticating server from DoS-attack.

The subject is filtering of RADIUS-packets with further their blocking in case of suspicious activity.

Keywords: DoS-attack, authenticating, RADIUS, packet filtering, blocking, security.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CHAP – Challenge Handshake Authentication Protocol

DDoS – Distributed Denial of Service

DoS – Denial of Service

IDS – Intrusion Detection System

IPS – Intrusion Prevention System

LAN – Local Area Network

NAC – Network Access Control

NAS – Network Access Server

PAP – Password Authentication Protocol

PPP – Point-to-Point Protocol

RADIUS – Remote Authentication Dial In User Service

RadSec – RADIUS Over TLS

RFC – Request for Comments

TCP – Transmission Control Protocol

TLS – Transport Layer Securit

UDP – User Datagram Protocol

VPN – Virtual Private Network

WLAN – Wireless Local Area Network

ВСТУП

Доступ до мережі Інтернет є важливою складовою роботи різних галузей бізнесу. Компаніям часто необхідно відстежувати які пристрої та користувачі автентифікуються та мають доступ до корпоративної мережі. Головна мета цього – попередження або активна протидія проти різних кібератак або ж проста заборона несанкціонованого доступу.

Для цього існують рішення контролю доступу до мережі, в рамках якого для проведення автентифікації та обліку користувачів використовується RADIUS-сервер. Дана технологія працює по протоколу RADIUS, за яким під час спілкування клієнт (точка доступу, тощо) та сервер обмінюються так званими RADIUS-повідомленнями різних типів, за якими пристрою або надається або забороняється до доступ до мережі.

При цьому також існує необхідність захисту даних рішень, а саме RADIUS-серверу, від різноманітних атак, у тому числі однієї з поширених – DoS атаки, тобто створення надлишкового трафіку у вигляді непотрібних чи деформованих RADIUS пакетів.

Робота є актуальною, адже підтримка адекватного функціонування корпоративної мережі є важливою, бо в результаті несправносте бізнес може втратити багато грошей або цінної інформації.

Об’єктом дослідження є система, що дозволяє захищати сервер автентифікації RADIUS від DoS атак. Предметом є фільтрація RADIUS-пакетів з подальшим їх блокуванням у випадку сумнівної активності.

Метою даної дисертації є дослідження та розробка системи для захисту сервера автентифікації RADIUS від DoS атак шляхом фільтрації пакетів. Завданням роботи є автоматизація захисту сервера RADIUS від DoS атак.

Методом дослідження в даній роботі є аналіз.

Розроблювана система має практичне значення в будь-якій системі, що автентифікує користувачів за допомогою серверу RADIUS.

На захист винесено рішення, що полягає в автоматизації методів захисту автентифікації RADIUS від DoS атак. Метод полягає у фільтрації пакетів автентифікації та обліку, використовуючи дані пристрою з пакету, та впровадженням порогового значення за певний період часу. Такий підхід не реалізується в проаналізованих аналогах, тому є відмінним від вже існуючих.

Магістерська дисертація складається з наступних розділів: вступ, розділ постановки задачі, розділ аналізу предметної області, розділ аналізу існуючих рішень, розділ огляду методів виявлення та захисту від DoS атак, розділ формування вимог, розділ опису архітектури, розділ реалізації системи, розділ стартапу, висновки, список використаних джерел з 23 найменувань, 1 додатку, графічна частина включає 8 креслеників формату А3.

У розділі 1 описано мету дисертації та поставлено її завдання.

У розділі 2 було проведено аналіз предметної області, а саме було описано системи контроля мережі, принцип роботи протоколу RADIUS та DoS атак.

У розділі 3 було досліджено існуючі аналоги системи, що розроблюється, проаналізовано їх недоліки та переваги та проведено їх порівняння.

У розділі 4 було розглянуто методи виявлення DoS атак та захисту від них.

У розділі 5 на основі переглянутих рішень було визначено функціональні та нефункціональні вимоги до майбутньої системи, описано сценарії її використання.

У розділі 6 описано архітектуру системи, алгоритм фільтрації пакетів та обґрунтовано вибір технологій для реалізації.

У розділі 7 описано реалізацію розроблюваної системи.

У розділі 8 описано стартап на основі розроблюваної системи.

1 ПОСТАНОВКА ЗАДАЧІ

Метою магістерської дисертації є дослідження та розробка системи для захисту сервера автентифікації RADIUS від DoS атак шляхом фільтрації пакетів.

Для досягнення поставленої мети буде вирішено наступні задачі:

- огляд методів захисту від DoS атак;
- проведення аналізу існуючих рішень;
- формування функціональних та нефункціональних вимог системи;
- розробка алгоритму фільтрації RADIUS-пакетів;
- розробка системи захисту сервера автентифікації RADIUS від DoS атак шляхом фільтрації пакетів.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

На сьогоднішній день Інтернет використовується усюди, та доступ до мережі є важливою складовою роботи різноманітних бізнесів – від заводів та фабрик до офісів.

Компаніям необхідно відслідковувати які пристрої та як саме вони під’єднуються до корпоративної мережі для захисту від несанкціонованого доступу та різноманітних кібер-атак.

Саме для рішення даної задачі існує контроль доступу до мережі (NAC, Network Access Server) [1], що є підходом до комп'ютерної безпеки, який намагається уніфікувати технологію захисту кінцевих точок, автентифікацію користувачів та забезпечення безпеки мережі. Рішення NAC стали надзвичайно цінним інструментом за останні роки, оскільки мобільні пристрої та Інтернет речей стали популярними в різних галузях світу. В рамках даного рішення для проведення автентифікації використовується сервер RADIUS, що дозволяє автентифіковувати пристрої в мережі. Але навіть для систем, що допомагають підтримувати безпеку у Інтернет просторі або при підключенні до нього, потрібен певний захист. Однією із можливих загроз – є DoS атака, що є нападом на систему з метою зробити її недоступною, що у випадку компаній абсолютно недопустимо, так як в певних галузях багато що залежить від з’єднання з Інтернетом і якщо система контролю доступу до мережі буде недоступна, то робота компанії може зупинитися на певний час, що принесе за собою можливі збитки для бізнесу.

2.1 Контроль доступу до мережі

Оскільки зараз організаціям доводиться враховувати експоненціальне зростання мобільних пристроїв, що отримують доступ до своїх мереж, та небезпеки, які вони несуть, надзвичайно важливо мати інструменти, які забезпечують видимість, контроль доступу та можливості дотримання вимог, необхідних для зміцнення інфраструктури безпеки мережі.

Корпоративна мережа може включати в себе наступні пристрої:

– концентратори з'єднують між собою кілька пристроїв локальної мережі (LAN, Local Area Network). Вони не виконують функції фільтрації та адресації пакетів;

– комутатори зазвичай виконують більш розумну роль, ніж концентратори. Ланцюги локальних мереж, як правило, підключаються за допомогою комутаторів. В основному, працюючи на рівні Data Link, вони читають заголовки пакетів і обробляють пакети належним чином. Як правило, комутатори можуть зчитувати апаратні адреси вхідних пакетів, щоб передавати їх до відповідного пункту призначення;

– маршрутизатори допомагають передавати пакети до місця призначення, позначаючи шлях через велику кількість взаємопов'язаних мережевих пристроїв. Вони видаляють пакети з вхідних кадрів, аналізують їх окремо та призначають IP-адреси;

– мости використовуються для з'єднання двох або більше хостів або сегментів мережі разом. Основна роль мостів у мережевій архітектурі - зберігання та пересилання кадрів між різними сегментами, які мост з'єднує;

– шлюзи зазвичай працюють на транспортному та сесійному рівнях моделі OSI. На транспортному рівні та вище є численні протоколи та стандарти від різних постачальників; шлюзи використовуються для боротьби з ними.

Для захисту корпоративних мереж можна використовувати наступні засоби:

– брандмауер, що є одною з перших ліній оборони в мережі, яка захищає одну мережу від іншої. Брандмауери можуть бути як окремими системами, так і частиною інших пристроїв, такі як маршрутизатори або сервери;

– система виявлення вторгнень (IDS, Intrusion Detection System) підвищує кібербезпеку, виявляючи в мережі хакера або шкідливе програмне забезпечення, щоб можна було негайно видалити його, для запобігання різним проблемам;

– система запобігання вторгненню (IPS, Intrusion Prevention System) є мережевим рішення безпеки, яке може не тільки виявляти зловмисників, але й перешкоджати їм успішному запуску будь-якої відомої атаки. Системи запобігання

проникненню поєднують у собі можливості брандмауерів та систем виявлення вторгнень;

– проксі-сервери виступають у ролі переговорів щодо запитів від клієнтського програмного забезпечення, що шукає ресурси з інших серверів. Клієнт підключається до проксі-сервера, запитуючи якусь послугу (наприклад, веб-сайт); проксі-сервер оцінює запит, а потім дозволяє або забороняє його. В організаціях проксі-сервери зазвичай використовуються для фільтрації трафіку та підвищення продуктивності;

– анти-DDoS-пристрої виявляють розподілені атаки відмови в обслуговуванні (DDoS, Distributed Denial of Service) на ранніх стадіях, поглинають обсяг трафіку та визначають джерело атаки;

– контроль доступу до мережі передбачає обмеження доступності мережевих ресурсів для пристроїв кінцевих точок, які відповідають потрібним політикам безпеки. Деякі рішення NAC можуть автоматично виправляти невідповідні вузли, щоб забезпечити їх безпеку до дозволу доступу. NAC є найбільш корисним, коли середовище користувача є досить статичним і може жорстко контролюватися, наприклад, підприємства та державні установи. Це може бути менш практичним у налаштуваннях із різноманітним набором користувачів та пристроїв, які часто змінюються, що є загальним у сферах освіти та охорони здоров’я.

В рішеннях контролю доступу до мережі зазвичай використовують RADIUS-орієнтовану серверну архітектуру, при якій RADIUS-сервер розгортається в дата-центрі. Різноманітні мережеві пристрої, такі як маршрутизатори, контролери бездротового доступу, або точки доступу, інтегруються з центральним RADIUS-сервером.

Головними перевагами такого підходу є наступні:

– лише один пристрій NAC для розгортання;

– лише один пристрій NAC для управління з часом;

– менша вартість апаратного забезпечення, якщо апаратне забезпечення розгорнуто;

– потенційно менші витрати на поточне обслуговування залежно від ліцензування.

Тим не менш, головною проблемою такого підходу є те, що необхідно інтегрувати RADIUS-сервер з кожним пристроєм мережі, що, при потребі, робить подальше редагування конфігурації цих пристроїв доволі важкою та незручною задачею.

2.2 Опис протоколу RADIUS

Для автентифікації і авторизації користувачів або організації їх обліку можна використовувати протокол RADIUS. Особливо, якщо мова йде про великі мережах, доступ до яких дозволений чималому числі людей.

Основні складові частини служби ідентифікації віддалених користувачів RADIUS описуються двома RFC від IETF: RFC 2865[2] під назвою Remote Authentication Dial-In User Service (RADIUS) у формі проекту стандарту і RFC 2866 [3] під назвою RADIUS Accounting в вигляді «інформаційного RFC». Спочатку концепція RADIUS полягала в забезпеченні віддаленого доступу через комутоване телефонне з'єднання. Згодом відділилися і інші області застосування цієї технології. До них відносяться сервери VPN (вони в більшості своїй підтримують RADIUS), а також точки доступу бездротових локальних мереж WLAN, і це далеко не все.

Концепція служби ідентифікації віддалених користувачів має на увазі, що клієнт RADIUS – зазвичай сервер доступу, сервер VPN або точка доступу бездротової локальної мережі – відсилає серверу RADIUS параметри доступу користувача (дані для автентифікації у мережі), а також параметри відповідного з'єднання. Для цього клієнт використовує спеціальний формат, так званий RADIUS-Message (повідомлення RADIUS). У відповідь сервер починає перевірку, в ході якої він автентифікує і авторизує запит клієнта RADIUS, а потім пересилає йому відповідь – RADIUS-Message-Response. Після цього клієнт передає на сервер RADIUS облікову інформацію.

Ще одна особливість – підтримка агентів RADIUS. Ці системи призначені виключно для забезпечення обміну повідомленнями RADIUS між клієнтами, серверами і іншими агентами. Звідси можна зробити висновок, що повідомлення ніколи не передаються безпосередньо від клієнта до сервера.

Самі по собі повідомлення RADIUS передаються в формі пакетів UDP. Причому інформація про автентифікації направляється на порт UDP з номером 1812. Деякі сервери доступу використовують, однак, порти 1645 (для повідомлень про автентифікації) або, відповідно, 1646, для обліку – вибір повинен визначатися рішенням адміністратора. В поле даних пакета UDP (так звана корисне навантаження) завжди поміщається тільки одне повідомлення RADIUS. Відповідно до RFC 2865 і RFC 2866 визначені наступні типи повідомлень:

Access-Request – “запит доступу". Запит клієнта RADIUS, з якого починається власне автентифікація і авторизація спроби доступу в мережу;

Access-Accept – “доступ дозволено". За допомогою цієї відповіді на запит доступу клієнту RADIUS повідомляється, що спроба з'єднання була успішно автентифікована і авторизована;

Access-Reject – “доступ не дозволено". Ця відповідь сервера RADIUS означає, що спроба доступу до мережі не вдалася. Таке можливо в тому випадку, якщо для користувача даних недостатньо для успішної автентифікації або доступ для користувача не авторизований;

Access-Challenge – “виклик запиту". Сервер RADIUS передає його у відповідь на запит доступу;

Accounting-Request – “запит обліку", який клієнт RADIUS відсилає для введення облікової інформації після отримання дозволу на доступ;

Accounting-Response – “відповідь обліку". Таким чином сервер RADIUS реагує на запит обліку і підтверджує факт обробки запиту обліку.

Повідомлення RADIUS завжди складається з заголовка і атрибутів, кожен з яких містить ту чи іншу інформацію про спробу доступу: наприклад, ім'я та пароль користувача, запитувані послуги і IP-адреса сервера доступу.

RADIUS може спільно працювати з різними протоколами автентифікації. Найбільш часто використовуються протокол автентифікації пароля (Password Authentication Protocol, PAP), протокол автентифікації з попереднім узгодженням (Challenge Handshake Authentication Protocol, CHAP), а також MS-CHAP (CHAP від Microsoft в першій версії або MS-CHAPv2 – у другій). Крім того, можливе застосування RADIUS разом з PPP, протоколом передачі «точка-точка» (Point-to-Point Protocol). Результати сеансу автентифікації між сервером доступу і чинним клієнтом передаються на сервер RADIUS, який їх потім засвідчує. Принцип роботи протоколу зображено на рисунку 2.1

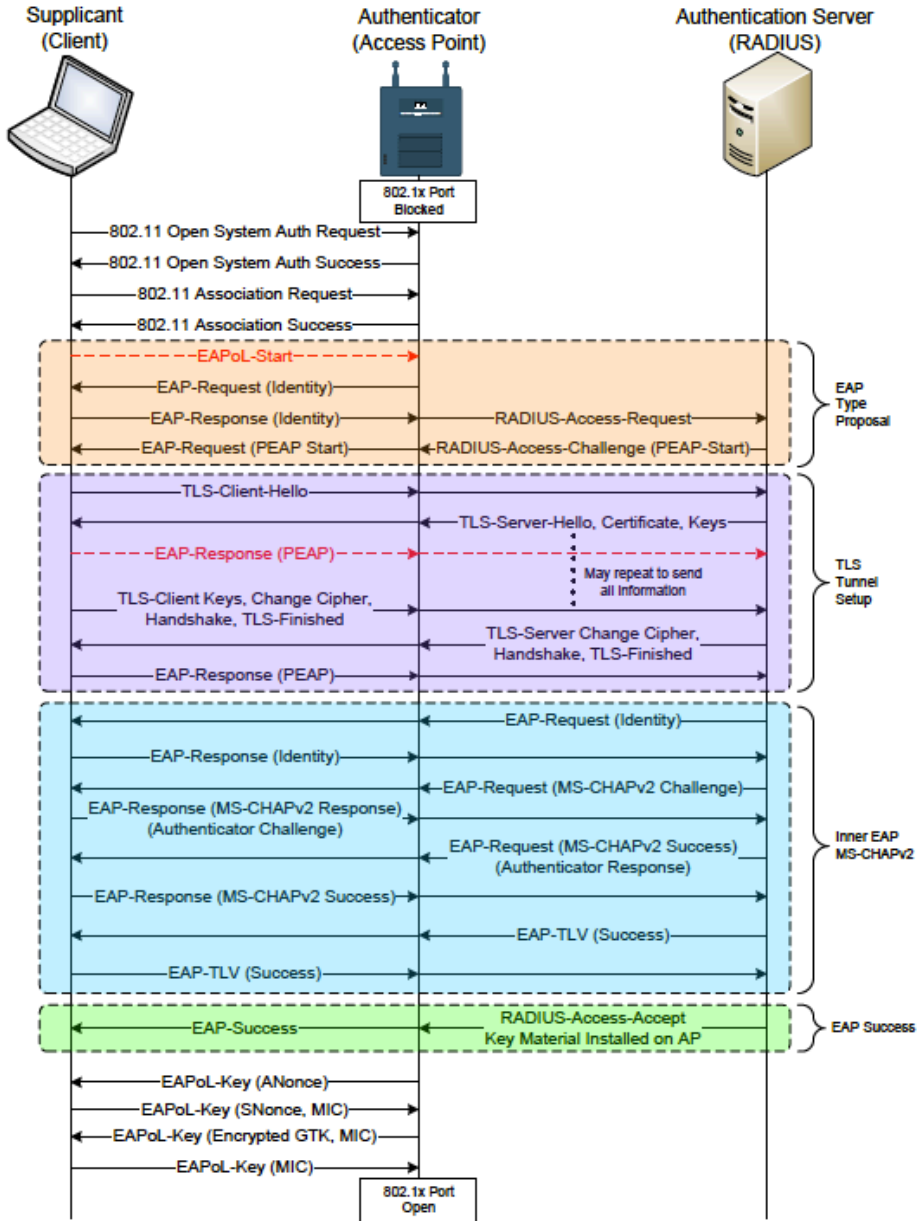

Рисунок 2.1 – Принцип роботи протоколу RADIUS

Для захисту повідомлень клієнт і сервер RADIUS володіють «shared secret» або, простіше кажучи, ключем. При цьому мова, як правило, йде про ланцюжок символів, наявні як на серверах, так і на клієнті RADIUS.

Також існує розширення даного протоколу – RadSec, відомий як RADIUS по TLS [4].

RADIUS по TLS призначений для забезпечення безпечної передачі запитів RADIUS за допомогою протоколу TLS. RadSec перенаправляє звичайний RADIUS-трафік на віддалені сервери RADIUS, підключені через TLS. RadSec дозволяє безпечно передавати дані автентифікації, авторизації та обліку RADIUS через ненадійні мережі.

RadSec використовує TLS у поєднанні з протоколом TCP. Цей транспортний профіль забезпечує більш надійний захист, ніж протокол UDP, який спочатку використовувався для передачі RADIUS. RADIUS по UDP, як було зазначено, шифрує загальний секретний пароль, використовуючи алгоритм MD5, який є вразливим до атак. RadSec, в свою чергу, зменшує ризик атак на MD5, обмінюючись корисними навантаженнями пакетів RADIUS через зашифрований тунель TLS. Схематично його роботу зображено на рисунку 2.2

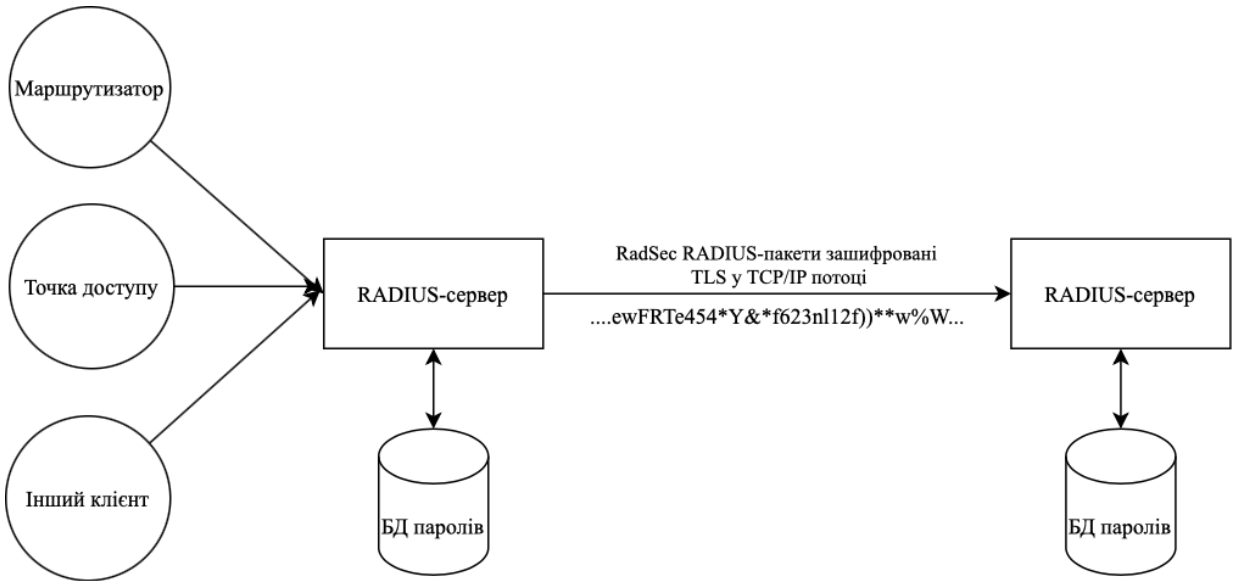

Рисунок 2.2 – Принцип роботи RADIUS по TLS

Зазвичай це реалізується в одній галузі: роумінговому середовищі. Коли пристрій переходить між мобільними або стільниковими даними на локальну точку доступу Wi-Fi, він або повинен від’єднатись і знову під’єднатися до Інтернету, або бути «переданим» мережами.

2.3 DoS атаки

DoS атака може бути проведена заповненням сервера деформованими, пошкодженими або непотрібними пакетами, в результаті чого сервер втрачає свою ресурсну потужність і як результат можливість спілкуватися зі своїми звичайними користувачами або обробляти регулярні запити. [5]

DoS атаки можуть бути різних типів:

- об'ємна атака – це атаки, при якій витрачається вся пропускна здатність мережі, тому авторизовані клієнти не можуть отримати ресурси. Це досягається шляхом заповнення мережевих пристроїв, таких як концентратори або комутатори, численними пакетами echo запитів/відповідей ICMP, завдяки чому витрачається вся пропускна здатність, і жоден інший клієнт не може підключитися до цільової мережі;
- заповлення Syn – атака, коли зловмисник компрометує кілька зомбі і одночасно заливає ціль декількома пакетами SYN. Ціль буде перевантажена запитами SYN і вона або перестає працювати, або її продуктивність різко знижується;
- атаки фрагментації – це атака, яка бореться проти здатності цілі збиратись знову. До цілі надсилаються численні фрагментовані пакети, що ускладнює їх повторне складання, тим самим, забороняючи доступ дійсним клієнтам;
- така виснаження стану TCP – зловмисник налаштовує та розриває TCP-з'єднання та перевантажує стабільні таблиці; тим самим викликаючи атаку DoS.
- атаки рівня програми – зловмисник використовує помилки програмування в застосунку, щоб викликати DoS. Це досягається шляхом надсилання численних запитів до цілі для вичерпання її ресурсів, щоб вона не змогла обслуговувати жодного дійсного клієнта;

– plashing – робиться шляхом нанесення повної шкоди апаратному забезпеченню шляхом надсилання шахрайських оновлень на апаратне забезпечення, роблячи їх повністю непридатними для використання. Єдиним рішенням є перевстановлення обладнання.

Вплив DoS атаки залежить від цілі. Орієнтація на конкретного клієнта може призвести до відмови в користуванні лише цьому користувачеві, але коли ціллю є RADIUS-сервер, жоден користувач не може бути автентифікований в мережі, і система контролю доступу до мережі не зможе адекватно працювати. Коли атакується RADIUS-сервер, репутація NAC продукту також потерпає. В результаті компанія може втратити частину своїх існуючих та потенційних клієнтів.

DoS атаки можуть відбуватися не навмисно, а через неправильну конфігурацію пристрою користувача чи клієнта RADIUS, який у результаті надсилає велику кількість непотрібних та деформованих пакетів. Але навіть у такому випадку необхідний захист від таких запитів.

2.4 Висновки розділу

В даному розділі було описано предметну область, а саме було зазначено що таке контроль доступу до мережі, протокол RADIUS та його розширення RadSec, що працює по протоколу TCP з TLS на відміну від звичайного принципу його роботи – по протоколу UDP. Було описано принцип роботи даного протоколу між клієнтами та серверами, а також типи RADIUS-пакетів. До того ж було зазначено що таке DoS атака, принцип її реалізації та можливі види.

3 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

3.1 Застосунок AirMagnet WiFi Analyzer

Fluke Networks AirMagnet WiFi Analyzer [6] є галузевим стандартом в області аудиту та усунення несправностей в корпоративних мережах WiFi. AirMagnet WiFi Analyzer дозволяє IT-персоналу швидко вирішувати проблеми кінцевих користувачів, одночасно автоматично виявляючи загрози безпеки та інші уразливості мережі. Рішення включає в себе єдиний в галузі набір активних засобів діагностики WLAN, що дозволяє мережевим адміністраторам легко перевіряти і діагностувати десятки поширених проблем продуктивності бездротових мереж, в тому числі проблеми підключення, конфлікти пристроїв і багатомаршрутних сигналів.

AirMagnet WiFi Analyzer містить повністю сумісний механізм створення звітів, який автоматично проектує зібрану інформацію про мережі на вимоги сумісності з політикою і нормативними актами.

Даний застосунок доступний в двох версіях:

WiFi Analyzer Express, що забезпечує основні функції аудиту та усунення несправностей мережі WiFi з можливістю переглядати пристрої, автоматично визначати типові проблеми і встановлювати фізичне місце розташування обраних пристроїв.

WiFi Analyzer PRO розширює можливості, представлені у версії Express, і додає безліч інших, з тим щоб забезпечити тестування WiFi, що вирішує практично будь-які питання продуктивності, безпеки і звітності.

Отже, за допомогою даної системи можна проводити аналіз пакетів з можливістю їх захвату, оцінку несправностей мережі та автоматичне находження нелегальних пристроїв тощо.

Але підтримка даного застосунку закінчилася, тому ліцензійно це програмне забезпечення не отримати, а якщо і знайти, то його ціна є доволі високою – 226 тис.

грн [7]. До того ж інтерфейс користувача, що зображено на рисунку 3.1, є доволі застарілим, важким для сприйняття, адже застосунок датується 2009 роком.

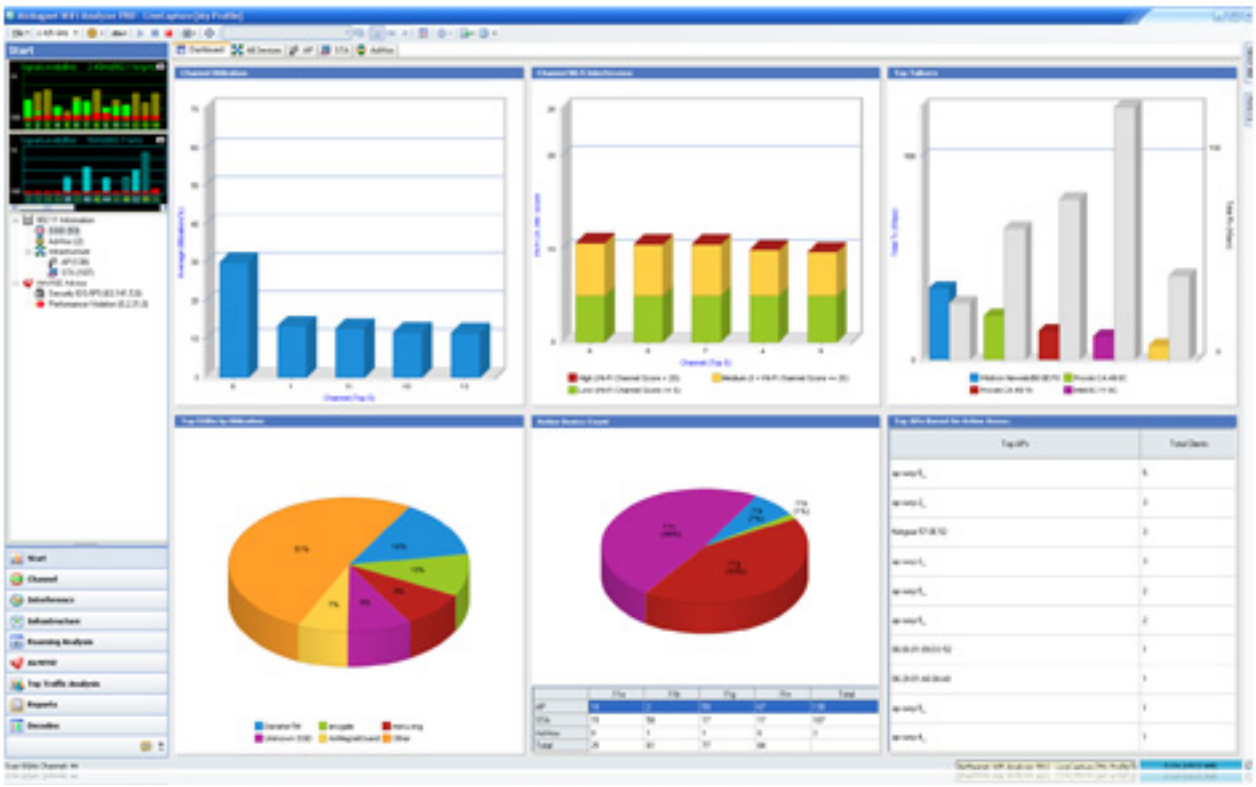

Рисунок 3.1 – Fluke Networks AirMagnet WiFi Analyzer

3.2 Застосунок Extreme AirDefense

Extreme AirDefense [8] спрощує захист, моніторинг та відповідність мереж бездротової локальної мережі. Даний застосунок постійно захищає мережу від зовнішніх загроз та повідомляє IT-персонал про напади, що дозволяє негайно реагувати.

Система Extreme AirDefense розгорнута як набір точок доступу, що служать датчиками для контролю ефіру разом з приладом безпеки. Прилад може бути розгорнутий як апаратний пристрій або віртуальний прилад. Датчики можна використовувати як спеціальні датчики, так і в режимі спільного використання радіо. Виділені датчики забезпечують вищий рівень безпеки завдяки підвищеній видимості. Існує два варіанти розгортання для спеціального зондування – вся точка доступу може бути виділена як датчик або подвійна точка радіо- або потрійний радіо доступу,

одна радіостанція точки доступу може бути призначена як датчик, а решта радіостанцій обслуговують трафік даних користувачів.

Інтерфейс цього застосунку зображено на рисунку 3.2.

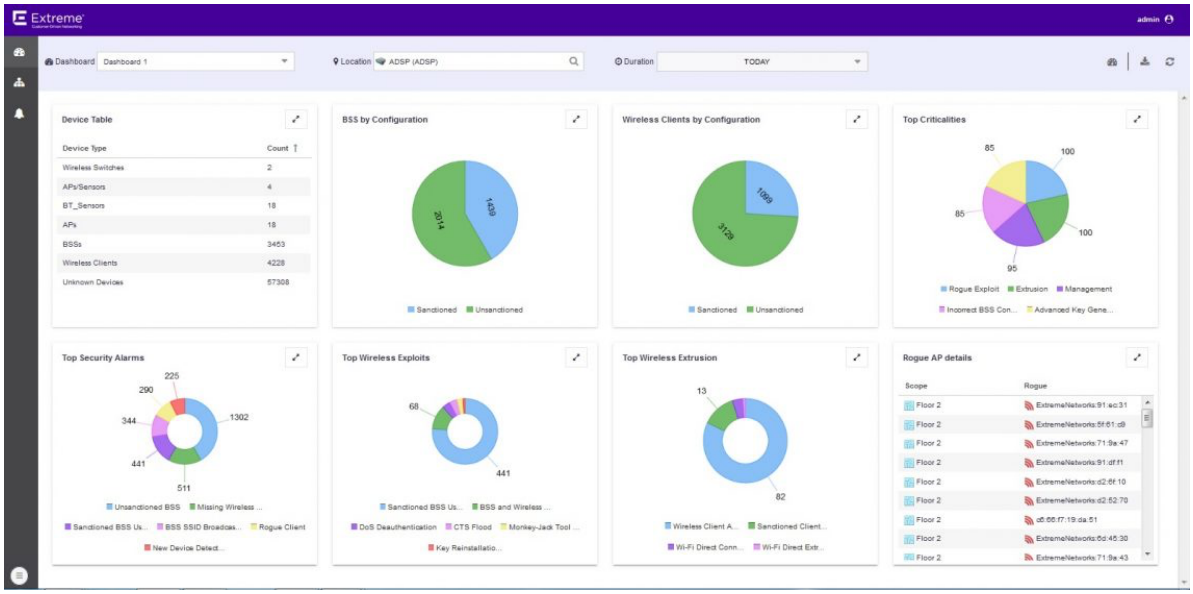

Рисунок 3.2 – Застосунок Extreme AirDefense

У режимі спільного використання радіо, точка доступу виділяє часовий зріз для функції зондування, використовуючи при цьому залишковий час для обслуговування трафіку даних. Екстремальні точки доступу, що працюють як датчики, підтримують стандарти 802.11a/b/g/n/ac/ax для сканування як діапазонів 2,4 ГГц, так і 5 ГГц і здатні слухати кілька потоків MIMO.

Хоча дана система є легко масштабованою і дозволяє підтримувати безпеку мережі в цілому, шляхом аналізу сигналів, для забезпечення захисту сервера RADIUS від DoS атак він має доволі надлишкову кількість інших функцій.

3.2 Застосунок Snort

Snort [9] – це передова система запобігання вторгнень з відкритим кодом (IPS) у світі. Snort IPS використовує ряд правил, які допомагають визначити шкідливу мережеву активність, і використовує ці правила для пошуку пакетів, які відповідають їм, та генерує попередження для користувачів.

Snort також можна розгорнути вбудовано, щоб зупинити ці пакети. Snort має три основних способи використання: як захват пакетів, такий як tcpdump, як реєстратор пакетів – що корисно для налагодження мережевого трафіку, або його можна використовувати як повноцінну систему запобігання вторгненню в мережу. До того ж Snort можна завантажити та налаштувати як для особистого, так і для ділового використання.

Даний застосунок підтримує створення та налаштування власних правил для роботи з трафіком та пакетами мережі, що є зручним, адже так можна вирішити певну вузьку проблему, що може не піддаватися рішенню стандартними засобами.

Наприклад, установлення порогу кількості запитів при якому можна вважати, що відбувається атака.

Тим не менш, використання цього програмного забезпечення потребує гарних навичок роботи з консоллю, бо він не має користувацького інтерфейсу, що і зображено на рисунку 3.3.

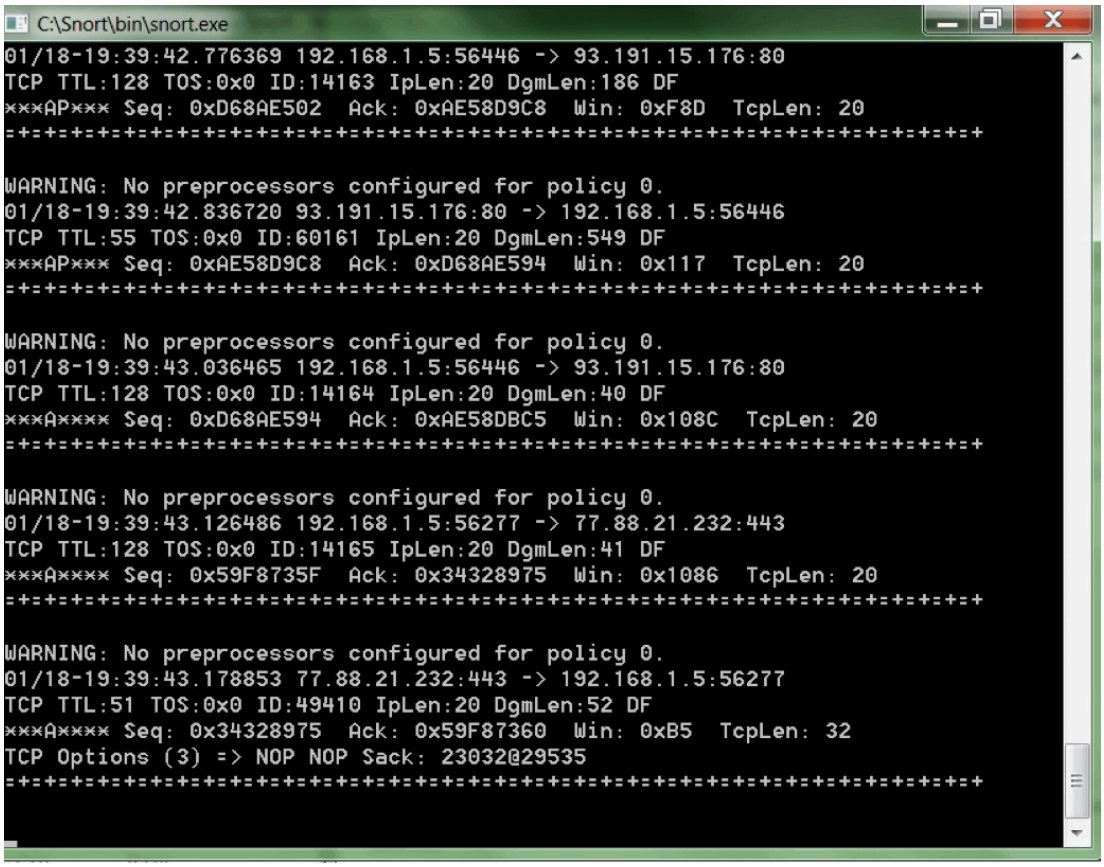

Рисунок 3.3 – Застосунок Snort

3.3 Порівняння проаналізованих застосунків

Для подальшого формування вимог до системи, що розроблюється, проведено порівняльну характеристику описаних вище аналогів. На основі порівняння характеристик цих застосунків було складено таблицю 3.1 для більш наглядного представлення інформації та порівняння.

Таблиця 3.1 – Порівняльний аналіз застосунків

Критерій оцінки	AirMagnet WiFi Analyzer	Extreme AirDefense	Snort
Аналіз пакетів	+	+	+
Управління пристроями	+	–	–
Блокування пристрою за потреби	+	–	+
Дружній до користувача інтерфейс	–	+	–
Підтримка користувацьких налаштувань	–	–	+

Таким чином, базуючись на даній таблиці можна проаналізувати якими характеристиками повинна володіти майбутня система для того, щоб вирішувати недоліки вже існуючих аналогів і тим самим мати над ними перевагу. При цьому вирішувати поставлену задачу і не мати надлишкових функцій як це було зазначено в описаних аналогах.

Головними характеристиками системи повинні бути наступні:

- аналіз пакетів;
- управління пристроями;
- блокування пристрою за потреби;
- дружній до користувача інтерфейс;
- підтримка користувацьких налаштувань.

Результати даного аналізу будуть використанні у подальшому при формування вимог до системи.

3.4 Висновки розділу

В даному розділі було розглянуто застосунки, що вже існують та виконують подібні функції до розроблюваної системи, а саме AirMagnet WiFi Analyzer, Extreme AirDefense та Extreme AirDefense.

Знайдені рішення подібні до розроблюваної системи лише кількома характеристиками та при цьому мають надлишкові функції, порівнюючи з необхідним функціоналом. Тому було проаналізовано їх характеристики з метою подальшого використання цих даних при формуванні вимог до майбутньої системи.

А саме в результаті аналізу було визначено, що одними з важливих для подібної системи функцій є наступними:

- аналіз пакетів;
- управління пристроями;
- блокування пристрою за потреби;
- дружній до користувача інтерфейс;
- підтримка користувацьких налаштувань.

4 ОГЛЯД МЕТОДІВ ВИЯВЛЕННЯ ТА ЗАХИСТУ ВІД DoS АТАК

4.1 Метод, що базується на порозі

Даний метод полягає в тому, що задається певна величина та її значення, наприклад, кількість запитів до серверу за хвилину. І в залежності від того було це значення перевищено чи ні відбувається певна дія.

В контексті використання RADIUS-серверу прикладом такого методу запобігання DoS атакам є просто блокування певного RADIUS-порту при досягненні певного ліміту запитів. Це є дуже швидкою та простою протидією надмірній кількості запитів, але при цьому блокується уся робота RADIUS-сервера на певному порті, тому пристрої, що хочуть автентифікуватися, не зможуть це зробити.

Отже, можна дійти до висновку, що по-перше, одиночний поріг призводить до частого хибно позитивного результату та по-друге, такий поріг важко визначити. При цьому він є простим в реалізації і виявляє проблему в режимі реального часу.

4.2 Метод, що базується на виділенні ознак

Даний метод полягає у виділенні певних рис, що при аналізі, так чи інакше, дозволяє отримати потрібний результат. Використання лише певних рис, підібраних відповідно до ситуації, допомагає ідентифікувати об’єкт точніше, ніж просто використовуючи одну характеристику, як, наприклад, у методі, що базується на порозі.

Може бути розширеним нейронними мережами, що зможуть автоматично підбирати необхідні значення зазначених рис в залежності від ситуації.

Головною перевагою такого методу є високий рівень виявлення та низький показник хибно позитивних результатів, а також надійність в різних мережевих сценаріях. Але при цьому потрібно виділити та обробити необхідні риси заздалегідь.

4.3 Фільтрація MAC-адрес

Одним з методів протидії DoS атакам є фільтрація по MAC-адресам на RADIUS-клієнті.

Адреса MAC унікальна для кожної мережевої карти, тому за допомогою фільтрації MAC-адрес можна дозволити або заблокувати доступ до мережі для визначених пристроїв використовуючи чорні та білі списки.

Фільтрацію по MAC у безпровідних мережах можна отримати шляхом сканування MAC (наприклад, через airodump-ng) та підміни власного MAC на один із підтверджених. Використання фільтрації за MAC-адресою може сприяти хибному почуттю безпеки.

Однак фільтрація за адресою MAC більш ефективна в провідних мережах, так як отримання довірених адрес для нападів є більш складними завданням. Фільтрація MAC також використовується у безпровідних корпоративних мережах з кількома точками доступу, щоб запобігти спілкуванню клієнта з іншим. Точка доступу може бути налаштована так, щоб дозволити клієнтам з'єднуватися лише із шлюзом за замовчуванням, але ніяк не з іншими клієнтами безпровідної мережі.

Недоліком такого підходу є те, що MAC адреса може бути підмінена (так званий MAC-spoofing), та необхідно вручну додавати небажані адреси до списку, адже більша частина мережевих пристроїв, таких як точки доступу, самостійно не можуть робити автоматичну фільтрацію MAC-адрес в режимі реального часу.

4.4 Фільтрація пакетів

Ще одним методом є фільтрація пакетів.

Загалом це техніка брандмауера, яка використовується для управління мережевим доступом шляхом моніторингу вихідних та вхідних пакетів і дозволяючи їм передавати або зупиняти на основі IP-адрес джерела та призначення, протоколів та портів.

Сам RADIUS протокол зазначає, що під час обміну пакетами на автентифікацію та облік між RADIUS-клієнтом та сервером передається певна інформація про пристрій та певні RADIUS-атрибути. Ця інформація може бути використана при аналізі пакетів під час процесу автентифікації або обліку.

Такий підхід дозволяє блокувати лише певні пристрої, які вважаються сумнівними, тим самим робота серверу не порушується і інші пристрої можуть бути автентифіковані не дивлячись на певні спроби DoS атак.

4.5 Висновки розділу

В даному розділі було розглянуто два типи методів виявлення DoS атак, а саме ті, що базуються на порозі, та ті, що базуються на виділенні ознак.

У порівнянні з пороговим виявленням, виявлення за ознаками зазвичай має кращі показники швидкості виявлення та є більш надійним.

Серед можливих методів захисту від DoS атак було описано принципи фільтрація по MAC-адресам, за якою доступ до мережі може бути дозволений чи заборонений лише певному переліку MAC-адрес, та фільтрація пакетів під час якої аналізується їх вміст та на основі цього аналізу пакети надсилаються далі або блокуються.

5 ФОРМУВАННЯ ВИМОГ

5.1 Функціональні вимоги

Функціональні вимоги – це перелік функцій, які повинна виконувати система, причому має бути вказано, як система реагує на ті чи інші вхідні дані, як вона поводиться в певних ситуаціях і т.д. Також у деяких випадках може бути вказано, що система не повинна робити. [10]

Функціональні вимоги визначають функціональність програмного забезпечення, що розробники повинні побудувати, щоб користувачі змогли виконати свої завдання в рамках бізнес-вимог.

Використовуючи порівняння характеристик існуючих аналогів в розділі 3.3, що враховувало їх переваги та недоліки, було створено перелік необхідних функціональних вимог до розроблюваної системи.

Вони є наступними:

- система повинна блокувати запити з пристроїв, що порушують зазначені правила (FR1);
- система повинна мати можливість налаштовувати поріг запитів та інтервал часу (FR2);
- система повинна мати можливість розблокування пристроїв адміністратором (FR3);
- система повинна мати можливість перегляду заблокованих пристроїв (FR4);
- система повинна мати можливість авторозблокування пристрою через зазначений період часу у випадку хибного блокування (FR5);
- система повинна мати можливість перегляду статистики запитів до серверу (FR6);
- система повинна мати можливість входу за логіном та паролем (FR7).

5.2 Нефункціональні вимоги

Нефункціональні вимоги описують характеристики системи і її оточення, а не її поведінку. Також може приводитися перелік обмежень, що накладаються на дії і функції, виконувані системою.

Нефункціональні вимоги не пов'язані безпосередньо з функціями, виконуваними системою. Вони пов'язані з такими інтеграційними властивостями системи, як надійність, час відповіді або розмір системи. Крім того, нефункціональні вимоги можуть визначати обмеження на систему, наприклад на пропускну здатність пристроїв введення-виведення, або формати даних, які використовуються в системному інтерфейсі. Такі вимоги описують цілі і атрибути якості.

Атрибути якості представляють собою додатковий опис функцій продукту, виражене через опис його характеристик, важливих для користувачів або розробників.

Серед нефункціональних вимог до майбутньої системи можна зазначити наступні:

- система повинна масштабуватися – при великій кількості запитів необхідна підтримка масштабування, щоб ресурси системи не перенавантажувалися;
- система не повинна затримувати процес автентифікації та обліку – процес фільтрації та налізу пакетів повинен відбуватися швидко та без затримок;
- система повинна мати дружній до користувача інтерфейс – для зручного та легкого використання системи користувачем;
- система не повинна видаляти або спотворювати дані у випадку неочікуваного завершення роботи;
- система повинна адекватно працювати при великій кількості запитів до сервера.

5.3 Сценарії використання системи

Сценарії використання системи описують реальні приклади того, як один або більше людей або організації, взаємодіють з системою. Вони описують кроки, події та\або дії, які відбуваються під час взаємодії. Сценарії використання можуть бути дуже детальними, з точним зазначенням того, як хтось працює з призначеним для користувача інтерфейсом, або з досить високим рівнем опису критичних бізнес-процесів, але не із зазначенням того, як вони виконуються.

Головними сценаріями використання розроблюваної системи є наступні:

- перегляд статистики;
- перегляд заблокованих пристроїв;
- налаштування правил авторозблокування;
- налаштування правил блокування;
- автентифікація в системі.

Діаграма сценарії використання системи зображено у додатку Б та далі кожен сценарій описано більш детально.

В розроблювальній системі передбачається лише один користувач – це системний адміністратор, який відповідає за працездатність сервера автентифікації RADIUS, роботу та архітектуру мережі загалом, тощо.

Функціями даного користувача є наступні, він може:

- переглядати заблоковані пристрої;
- переглядати статистику щодо заблокованих пристроїв та активність запитів;
- розблоковувати пристрій;
- налаштовувати правила для блокування;
- налаштовувати правила для авторозблокування;
- бути автентифікований у розроблюваній системі.

На кожну з зазначених функцій є детально описаний свій сценарій користування системою.

Одним з головних сценарії використання для системного адміністратора в розроблюваній системі є перегляд заблокованих системою пристроїв.

Сценарій дозволяє адміністратору мати візуальне представлення у вигляді таблиці які пристрої та коли саме були заблоковані системо. з можливістю подальшого менеджменту цих пристроїв , а саме їх розблокування.

У таблиці 5.1 представлено опис зазначеного сценарію для перегляду заблокованих системою пристрої.

Таблиця 5.1 – Опис сценарію використання «Перегляд заблокованих пристроїв»

Назва	Перегляд заблокованих пристроїв
Ідентифікатор	UC1
Актор	Системний адміністратор
Частота використання	Кожен раз при потребі переглянути заблоковані пристрої
Тригери	Обрання сторінки заблокованих пристроїв
Передумови	Системний адміністратор автентифікований у системі; він хоче переглянути інформацію щодо заблокованих пристроїв
Постумови	Користувач переглядає список заблокованих пристроїв
Основний шлях	1. Обрати сторінку заблокованих пристроїв
Альтернативний шлях	Відсутній
Помилки	Відсутні

Наступним сценарієм використання для системного адміністратора є перегляд статистики по заблокованим системою пристроям та запитам, що проходять через систему.

Функція в основі даного сценарію передбачає, що адміністратор має можливість отримати статистку по тому які пристрої, як часто, на яку точку доступу тощо, намагалися під’єднатися, але були заблоковані через сумнівну активність. А також можливість візуально бачити інформацію щодо кількості запитів, які система оброблює.

Статистика представлена за допомогою графіків для кращого візуального сприйняття.

У таблиці 5.2 представлено опис даного сценарію, коли системний адміністратор хоче переглянути статистику щодо заблокованих системою пристроїв або ж активність в мережі.

Таблиця 5.2 – Опис сценарію використання «Перегляд статистики»

Назва	Перегляд статистики
Ідентифікатор	UC2
Актор	Системний адміністратор
Частота використання	Кожен раз при вході до системи
Тригери	Обрання сторінки зі статистикою
Передумови	Системний адміністратор автентифікований у системі; Він хоче переглянути статистику використання системи
Постумови	Системний адміністратор переглядає статистику використання системи
Основний шлях	1. Обрати сторінку статистики 2. Обрати потрібний тип інформації для представлення
Альтернативний шлях	Відсутній
Помилки	Відсутні

В рамках перегляду заблокованих системою пристроїв системний адміністратор має можливість розблокувати один або декілька одночасно пристроїв, щоб дозволити їм подальше підключення до мережі.

У таблиці 5.3 представлено опис сценарію, коли системний адміністратор хоче розблокувати вручну один або декілька заблокованих пристроїв.

Таблиця 5.3 – Опис сценарію використання «Розблокування заблокованого пристрою»

Назва	Розблокування заблокованого пристрою
Ідентифікатор	UC3
Актор	Системний адміністратор
Частота використання	Кожен раз при потребі розблокувати пристрій
Тригери	Обрання функції розблокування на екрані перегляду заблокованих пристроїв
Передумови	Системний адміністратор автентифікований; переглядає інформацію щодо заблокованих пристроїв; хоче розблокувати певний з них
Постумови	Системний адміністратор розблоковує потрібний пристрій, який зникає з списку заблокованих
Основний шлях	1. Обрати сторінку заблокованих пристроїв 2. Натиснути на розблокування на записі заблокованого пристрою
Альтернативний шлях	1. Обрати сторінку заблокованих пристроїв 2. Обрати необхідні пристроїв в списку 3. Натиснути на розблокування пристроїв
Помилки	Відсутні

При використанні системи системний адміністратор також має можливість налаштовувати те за якими параметрами система буде оцінювати чи є активність пристрою сумнівна для подальшого його блокування.

Серед цих налаштувань є і встановлення числових значень, таких як граничне значення кількості запитів за період часу, сам період за який вимірювати запити, так і використання власних користувацьких правил для отримання та аналізу інформації про пристрій.

У таблиці 5.4 представлено опис сценарію, коли системний адміністратор хоче налаштувати правила для блокування.

Таблиця 5.4 – Опис сценарію використання «Налаштування правил блокування»

Назва	Налаштування правил блокування
Ідентифікатор	UC4
Актор	Системний адміністратор
Частота використання	Кожен раз при потребі змінити налаштування блокування у системі
Тригери	Обрання сторінки з налаштуваннями
Передумови	Системний адміністратор автентифікований; він переглядає налаштування системи він хоче їх змінити
Постумови	Системний адміністратор змінює налаштування системи
Основний шлях	1. Обрати сторінку налаштувань 2. Змінити потрібні налаштування
Альтернативний шлях	Відсутній
Помилки	1. Введення некоректних параметрів

У системі передбачена можливість авторозблокування заблокованих системою пристроїв у випадку, якщо вони були помилково були ідентифіковані системою як ті, що мають сумнівну активність в мережі.

Це можливий випадок, якщо пристрій був неправильно конфігуровано.

Системний адміністратор може налаштовувати період після якого пристрій, що було заблоковано системою лише один раз попередньо, буде розблокованим автоматично. Дана функція може бути увімкнена або ні.

У таблиці 5.5 представлено опис сценарію, коли системний адміністратор хоче налаштувати правила для авторозблокування пристроїв через певний час у випадку хибного блокування.

Таблиця 5.5 – Опис сценарію використання «Налаштування правил авторозблокування»

Назва	Налаштування правил авторозблокування
Ідентифікатор	UC5
Актор	Системний адміністратор
Частота використання	Кожен раз при потребі змінити налаштування авторозблокування пристрою
Тригери	Обрання сторінки з налаштуваннями
Передумови	Системний адміністратор автентифікований; він переглядає налаштування авторозблокування у системі; він хоче їх змінити
Постумови	Системний адміністратор розблоковує потрібний пристрій, який зникає з списку заблокованих
Основний шлях	1. Обрати сторінку налаштувань 2. Змінити налаштування авторозблокування
Альтернативний шлях	Відсутній
Помилки	1. Введення некоректних параметрів

Для користування можливостями розроблюваної системи, системний адміністратор повинен бути автентифікований та авторизований у ній, адже система має безпосередній контроль над тим які пристрої мають та не мають можливість під’єднуватися до корпоративної мережі.

Так як дана інформація не є тою, що може бути легко доступною та дозволеною для перегляду людьми, що не мають до роботи з ними безпосереднього відношення, такі як звичайні працівники компаній або користувачі, що використовують гостьовий доступ до мережі, то доступ до системи є лише за логіном та паролем.

У таблиці 5.6 представлено опис сценарію, коли системний адміністратор хоче зайти у систему, тобто бути у ній автентифікованим.

Таблиця 5.6 – Опис сценарію використання «Автентифікація в системі»

Назва	Автентифікація в системі
Ідентифікатор	UC6
Актор	Системний адміністратор
Частота використання	Кожен раз при вході у систему
Тригери	Вхід до системи
Передумови	Системний адміністратор відкрив застосунок
Постумови	Системний адміністратор автентифікований у системі; він потрапляє на головну сторінку застосунку
Основний шлях	1. Відкрити головну сторінку застосунку 2. Ввести логін та пароль для входу 3. Натиснути кнопку входу
Альтернативний шлях	Відсутній
Помилки	Відсутні

5.4 Матриця відповідності функціональних вимог до сценаріїв використання

Матриця відповідності вимог – це двовимірна таблиця, що містить відповідність функціональних вимог (FR) системи та сценаріїв користування нею (UC). У заголовках стовпчиків таблиці розташовані сценарії користування, а в заголовках рядків – вимоги. На перетині – відмітка, що означає, що вимога поточного стовпчика покрита сценарієм користування поточного рядка.

Таблиця дає візуальне відображення двох параметрів:

- наявність в системі вимог, які ще не покриті (якщо у вимоги немає жодного перетину з сценаріями користування);
- чи є в системі надмірне тестування (якщо вимоги мають кілька перетинів).

Дана матриця зображена у таблиці 5.7 де значенням FR1-7 відповідають функціональні вимоги відповідно до підрозділу 5.1, а UC – сценарії використання, що описані в підрозділі 5.3.

Таблиця 5.7 – Матриця відповідності вимог

	UC1	UC2	UC3	UC4	UC5	UC6
FR1	+					
FR2				+		
FR3			+			
FR4	+					
FR5					+	
FR6		+				
FR7						+

З даної матриці відповідності видно, що встановлені функціональні вимоги до системи покриті описані сценаріями користування системою та відсутнє надмірне тестування системи.

5.5 Висновки розділу

В даному розділі було розроблено вимоги до майбутньої системи, а саме функціональні, що зазначають перелік функцій, які система повинна виконувати, та нефункціональні, тобто фі, що описують її характеристики та можливі обмеження на функції системи.

На основі функціональних вимог було створено діаграму сценаріїв використання розроблюваної системи.

Отриману діаграму, за якою у системі є лише один користувач і це – системний адміністратор, було описано. Головними функціями цього користувача є:

- перегляд заблокованих пристроїв;
- перегляд статистики щодо заблокованих пристроїв та активність запитів;
- розблокування пристроїв;
- налаштовування правил для блокування;
- налаштовування правил для авторозблокування;
- автентифікація у розроблюваній системі.

Кожен з зазначених у діаграмі сценаріїв було детально описано.

Використовуючи описану у розділі інформацію було побудовано матрицю відповідності вимог до сценаріїв використання за якою було видно, що всі функціональні вимоги покриті сценаріями використання і при цьому немає надмірного тестування.

6 ОПИС АРХІТЕКТУРИ СИСТЕМИ

6.1 Вибір та обґрунтування технологій

6.1.1 Платформа .NET

Платформа .NET Framework – це середовище виконання, що управляє застосунками, призначеними для .NET Framework. Вона складається з середовища CLR, яка надає інструменти управління пам'яттю і інші служби системи, та великої бібліотеки класів, що дозволяє програмістам використовувати стійкий, надійний код у всіх основних областях розробки застосунків. [11]

C# – це мова програмування, призначена для розробки найрізноманітніших застосунків, призначених для виконання в середовищі .NET Framework. Мова C# є простою, типобезпечною і об'єктно-орієнтованою. Мова була розроблена з урахуванням сильних і слабких особливостей інших мов, зокрема Java і C++. Завдяки безлічі нововведень C# забезпечує можливість швидкої розробки застосунків, але при цьому зберігає виразність і елегантність, властиві мовам C.

Ключовими особливостями мови C# є наступні:

- компонентна орієнтованість;
- код зібраний воєдино (декларації і реалізації об'єднані разом);
- уніфікована система типів і їх безпечність;
- автоматична і мануальна робота за пам'яттю;
- використання єдиної бібліотеки класів – CLR.

В якості веб-фреймворку було вирішено використовувати .NET Core. У цієї технології є наступні переваги у порівняні з .NET Framework:

- є крос-платформною, а тому може працювати на різних операційних системах;
- є швидкою та здатна до масштабування;
- є новою та більш сучасною технологією;
- є доволі легкою та модульною.

Тим не менш у даного фреймворку є й свої певні недоліки:

- він є досить «молодим» та ще розвивається;

- певні бібліотеки не підтримують дану технологію;
- деякі функції .NET Framework ще не підтримуються;
- він не є повністю протестованим.

Не дивлячись на те, що фреймворку було впроваджено досить недавно, можна зробити висновок, що дана технологія є більш сучасною і краще підходить для розробки сучасних веб-застосунків.

Так як раніше було зазначено, що система буде розроблюватися з використанням архітектурного стилю REST, то .NET Core також підходить, тому що підтримує створення сервісів REST, також відомих як Web API.

6.1.2 Мова програмування C++

C++ це компільована строго типізована мова програмування загального призначення. Вона підтримує різні парадигми програмування: процедурну, узагальнену, функціональну; але найбільшу увагу приділено підтримці об'єктно-орієнтованого програмування.

Нововведеннями C ++ в порівнянні з C є:

- підтримка об'єктно-орієнтованого програмування через класи. C ++ надає всі чотири можливості ООП - абстракцію, інкапсуляцію, успадкування (в тому числі і множинне) і поліморфізм.
- підтримка узагальненого програмування через шаблони функцій і класів;
- стандартна бібліотека C ++ складається зі стандартної бібліотеки C (з деякими модифікаціями) і бібліотеки шаблонів (Standard Template Library, STL), яка надає широкий набір узагальнених контейнерів і алгоритмів;
- додаткові типи даних;
- обробка виключень;
- віртуальні функції;
- простору імен;
- вбудовуються (inline) функції;
- перевантаження (overloading) операторів;

- перевантаження імен функцій;
- посилання і оператори управління вільно розподіленою пам'яттю.

6.1.3 Сервер автентифікації FreeRADIUS

FreeRADIUS – це RADIUS-сервер з відкритим вихідним кодом. [12] Він є альтернативою інших комерційних RADIUS-серверів, оскільки він модульний і функціональний на сьогоднішній день. Крім того, він входить в п'ятірку RADIUS-серверів світу з точки зору розгортання і кількості користувачів, яких цей сервер авторизує щодня.

Може працювати на вбудованих системах з невеликою кількістю пам'яті, обслуговуючи кілька мільйонів користувачів. FreeRADIUS швидкий, гнучкий, легкий в налаштуванні, а також підтримує більше протоколів автентифікації, ніж більшість комерційних серверів. В даний час FreeRADIUS використовується як основа для розробки комерційних RADIUS-серверів.

Популярність FreeRADIUS можна пояснити безліччю додаткових переваг, які він пропонує, набагато перевищуючи ті, що знаходяться в широкому спектрі інших серверів RADIUS.

Даний сервер заснований на багатофункціональному, модульному та масштабованому протоколі проектування, який забезпечує наступні переваги та переваги мережевих адміністраторів:

Особливістю FreeRADIUS є те, що він підтримує більше типів автентифікації, ніж будь-який інший сервер із відкритим кодом. Наприклад, FreeRADIUS – єдиний сервер RADIUS із відкритим кодом, який підтримує розширюваний протокол автентифікації (EAP). Він також є єдиним RADIUS- сервером (комерційним або з відкритим кодом), який підтримує «віртуальні сервери». Використання віртуальних серверів означає спрощення складних реалізацій, а постійні витрати на підтримку та обслуговування адміністраторів мережі значно зменшуються; таким чином, здатність FreeRADIUS підтримувати віртуальні сервери дає йому величезну перевагу над конкурентами.

Модульна конструкція дозволяє легко його зрозуміти. Модульний інтерфейс також спрощує додавання або видалення модулів. Наприклад, якщо функція не потрібна для певної конфігурації, модуль легко видаляється. Видалення модуля не впливає на продуктивність сервера, використання пам’яті та безпеку. Ця гнучкість дозволяє серверу працювати на платформах, починаючи від вбудованих систем і закінчуючи багатоядерними машинами з гігабайтами оперативної пам'яті.

Наступною його перевагою є те, що один сервер RADIUS може легко перейти від обробки одного запиту кожні кілька секунд до обробки тисяч запитів в секунду, просто переналаштувавши кілька параметрів за замовчуванням. Багато великих організацій (ті, що мають понад 10 мільйонів клієнтів) залежать від FreeRADIUS для своїх потреб AAA. Хоча багато комерційних продавців пропонують різні версії свого програмного забезпечення для задоволення різних потреб, остання версія FreeRADIUS потрібна для отримання кращої продуктивності, більшої сфери, більшої кількості клієнтів RADIUS та багатьох інших функцій, без необхідності купувати додаткові ліцензії на продукт.

6.1.4 Хмарна платформа Microsoft Azure

На сьогодні хмарні технології є одною з найбільш популярних тем та напрямків розвитку в IT. Під поняттям хмарних технологій мається на увазі надання користувачу ресурсів та потужностей у вигляді інтернет-сервісу. Головна причина актуальності хмарних обчислень є те, що вони потребують менше витрат, мають кращі можливості до масштабування та гнучку архітектуру інформаційних технологій.

Перехід від стандартної локальної інфраструктури до хмари значно зменшує капітальні витрати не тільки на обладнання, а й на експлуатаційні витрати зв’язані з технічним обслуговуванням та дає змогу використовувати необхідні ресурси по мірі необхідності. [13]

Microsoft Azure є одним із переліку хмарних рішень, що надає такі можливості.

Microsoft Azure – це хмарна платформа з широким переліком ресурсів та послуг, які дозволяють полегшити роботу з ресурсами, шляхом швидкого створення, розгортання і керування сервісами.

Дане хмарне рішення включає в собі дві моделі — платформи як сервісу (PaaS) та інфраструктури як сервісу (IaaS) і поширюється за принципом «Pay only for what you use», за яким оплачуються лише ті послуги та ресурси, що використовуються. При цьому Azure також підтримує гібридний формат інфраструктури, що надає користувачу засоби для розширення можливостей зберігання, архівування та відновлення даних в максимально ефективному та економічному вигляді.

Можливості даної платформи постійно розширюються та забезпечують рішення більшої частини задач і сервісів, що користувачі вирішували та використовували за допомогою локальної інфраструктури.

Хмарна платформа Microsoft Azure надає велику кількість послуг та сервісів на своїх центрах обробки даних, які тісно інтегровані між собою і вважаються одним великим рішенням для вирішення будь-якої задачі. Цей факт робить дану популярнішою серед інших.

6.1.5 Фреймворк React.js

React – це бібліотека JavaScript, розроблена Facebook в 2013 році, яка відмінно підходить для створення сучасних односторінкових застосунків будь-якого розміру і масштабу. [14]

Перевагами React.js є наступні:

- легко вивчити, завдяки простому дизайну, використання JSX (HTML-подібний синтаксис) для шаблонів і дуже докладної документації. Розробники витрачають більше часу на написання сучасного JavaScript і менше турбуються про код, специфічному для фреймворка;
- дуже швидка, завдяки реалізації React Virtual DOM і різним оптимізаціям рендеринга;

– відмінна підтримка рендеринга на стороні сервера, що робить його потужною платформою для контент-орієнтованих застосунків;

– прив'язка даних є односторонньою, що означає менше небажаних побічних ефектів;

– Redux, найпопулярніша платформа для управління станом застосунків в React, її легко вчити і використовувати;

– React реалізує концепції функціонального програмування, створюючи простий в тестуванні і багаторазово використовуваний код;

– застосунки можуть бути створені за допомогою TypeScript або Facebook's Flow, що мають вбудовану підтримку JSX;

– перехід між версіями, як правило, дуже простий: Facebook надає «кодові модулі» для автоматизації більшої частини процесу;

– навички, отримані в React, можуть бути застосовані до розробки на React Native.

Тим не менш у нього є і недоліки, а саме:

– React не однозначний і залишає розробникам можливість обрати кращий спосіб розвитку;

– спільнота ділиться по способам написання CSS в React, які поділяються на традиційні таблиці стилів (CSS Modules) і CSS-in-JS (тобто Emotion і Styled Components);

– React відходить від компонентів на основі класів, що може стати перешкодою для розробників, яким більш комфортно працювати з об'єктно-орієнтованим програмуванням (ООП);

– змішування шаблонів з логікою (JSX) може збити з пантелику деяких розробників при перших знайомствах з React.

6.2 Опис архітектури компонентів системи

Головними компонентами в даній системі є власне сам RADIUS-сервер, мережевий екран між RADIUS-клієнтом (наприклад, точкою доступу) та сервером

автентифікації, зовнішній сервер для обробки даних з отриманих запитів та клієнтська частина для зручного адміністрування роботи системи. Структурна схема системи зображені на схемі В. Детально опишемо кожен з них.

Одним з головних компонентів системи є власне система серверу для захисту від DoS атак, що включає в себе мережевий екран та сервер системи захисту. Цю частину зображено на рисунку 6.1.

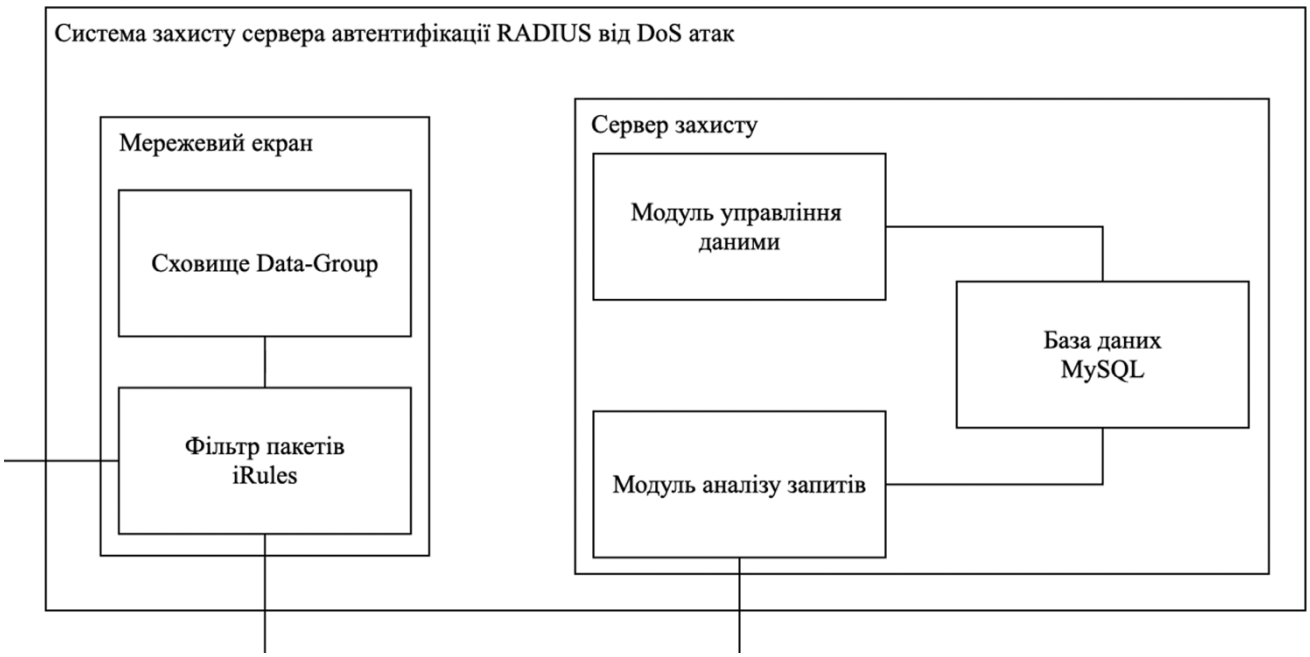

Рисунок 6.1 – Модуль «Система захисту сервера автентифікації RADIUS від DoS атак»

Мережевий екран в свою чергу має два наступні модулі: фільтр пакетів iRules та сховище Data-Group.

У фільтрі пакетів iRules з отримані від серверу мережевого доступу пакетів дістають необхідні дані для фільтрації та перевіряється наявність їх комбінації у сховищі Data-Group. Якщо дана комбінація присутня, то пакети з таким вмістом відхиляються. сховищі Data-Group являє собою своєрідну просту базу даних, яка зберігає дані у вигляді списку.

Модуль серверу системи захисту складається з таких модулів як модуль аналізу запитів, модуля управління даними та база даних MySQL.

Модуль аналізу даних отримує дані про автентифікацію з сервера автентифікації та оброблює їх на предмет перевищення ліміту на кількість запитів за період часу.

Модуль управління даними відповідає за роботу з даними про заблоковані пристрої з клієнтського інтерфейсу. Тобто системний адміністратор може зайти на веб-сторінку та переглянути статистику та заблоковані пристрої, розблокувати їх або змінити налаштування системи.

Даний модуль представлений як клієнт-серверна трирівнева архітектура, тобто є три рівні:

- рівень презентації: користувач працює безпосередньо з цим рівнем;
- рівень бізнес логіки: на даному рівні знаходиться уся бізнес-логіка, а саме обробка даних та опрацювання запитів від клієнта;
- рівень даних: цим рівнем представляється база даних, з якою пов’язана система і яка надає цій системі доступ до даних.

Перші два рівні, тобто рівень презентації та бізнес-логіки є частиною даного модуля, а рівень даних належить до модуля, що описано дані

Модуль управління даними та модуль аналізу даних спілкуються з базою даних MySQL, так як вся інформація про налаштування системи та заблоковані пристрої зберігається у ній, а ці два модулі використовують ці дані.

Наступний модуль на діаграмі є модуль RADIUS-сервера, він включає в себе ще два модулі: сервер автентифікації FreeRADIUS та фонову програму-процес.

Сервер автентифікації відповідає за обробку запитів на автентифікацію та облік від сервера доступу до мережі, які були пропущені мережевим екраном.

Інформація про оброблені запити потрапляє до наступного модулю – фонової програми-процес, що направляє їх далі на обробку до хмарних сервісів Azure.

Описаний модуль RADIUS-серверу зображено на рисунку 6.2.

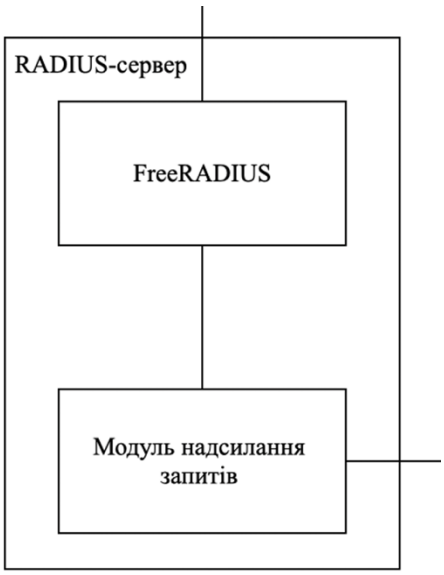

Рисунок 6.2 – Модуль «RADIUS-сервера»

Модуль, що відповідає за обробку даних в Azure зображено на рисунку 6.3.

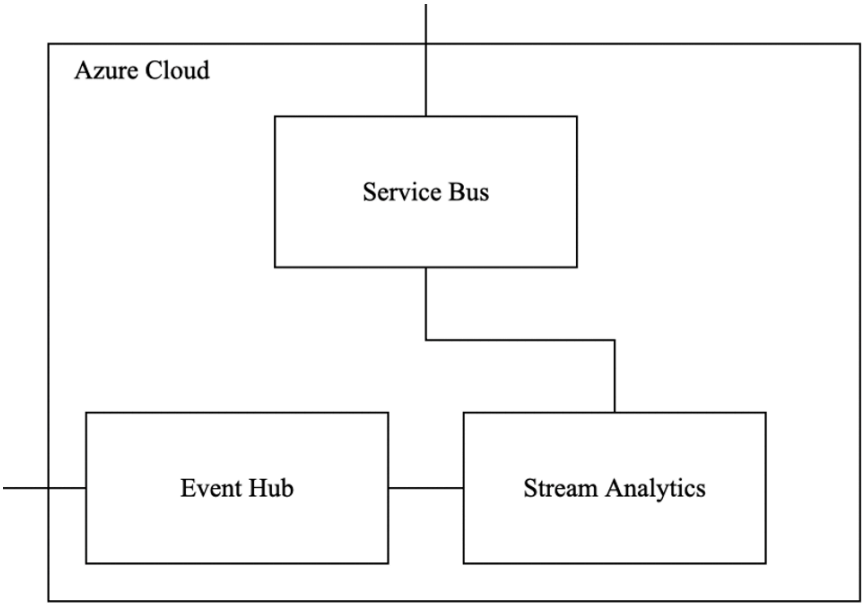

Рисунок 6.3 – Модуль «Azure Cloud»

Тут дані, що були отримані з сервера автентифікації спочатку направляються до модулю Event Hub, звідки їх отримує наступний модуль Stream Analytics для першого етапу аналізу – агрегації потрібних даних. Результат аналізу отриманих даних надсилається до модулю Service Bus, звідки в свою чергу модуль аналізу даних, що було описано раніше, їх отримує та оброблює.

6.3 Опис роботи мережевого екрану

Далі розглянемо принцип роботи системи з трафіком у мережевому екрані у двох випадках: простому UDP RADIUS трафіку та RADIUS через TLS.

Перший випадок є тривіальним і не потребує особливо важких налаштувань, адже UDP пакети не є зашифрованими, тому отримати інформацію з них дуже легко.

Принцип роботи мережевого екрану у випадку UDP зображено на рисунку 6.4.

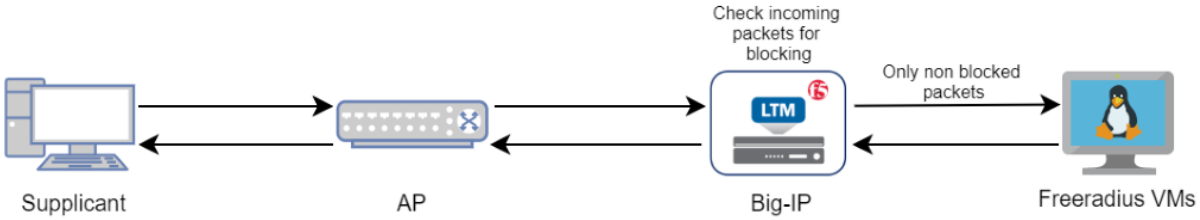

Рисунок 6.4– Схема принципу роботи з мережевим екраном у випадку UDP

У випадку RADIUS через TLS, як вже було зазначено, трафік, що проходить між RADIUS-клієнтом та сервером є зашифрованим. У випадку з мережевим екраном зашифрований трафік просто проходить через нього в такому ж самому вигляді, тобто на цьому проміжному етапі отримати доступ до даних, що знаходяться в тунелі неможливо. Принцип роботи цього випадку зображено на рисунку 6.5.

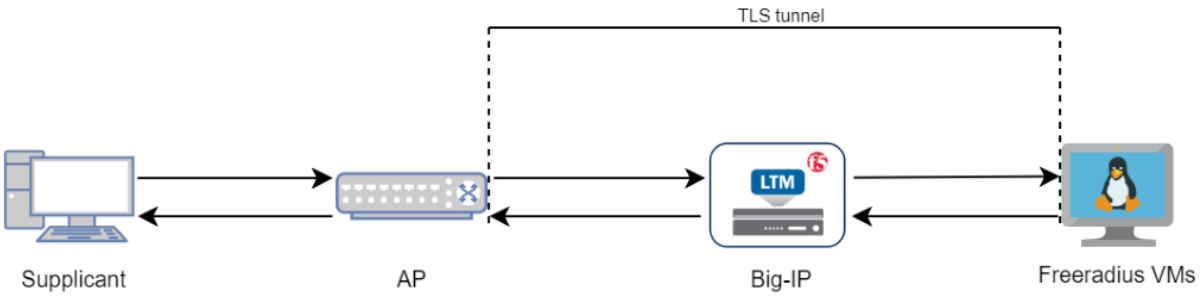

Рисунок 6.5 – Схема принципу роботи з мережевим екраном у випадку RADIUS через TLS трафіку

Але для коректної фільтрації пакетів необхідно спочатку розшифрувати дані пакета. Це може бути зроблено за допомогою так званого SSL Bridging.

SSL Bridging [15] – це процес, коли пристрій, що, як правило, розташований на межі мережі, розшифровує SSL трафік, а потім повторно шифрує його, перш ніж відправляти на веб-сервер. SSL Bridging може бути корисним, коли граничний пристрій виконує глибоку перевірку пакетів, щоб переконатись, що вміст зашифрованої SSL-передачі безпечний, або якщо є проблеми з безпекою щодо незашифрованого трафіку, що обходить внутрішню мережу.

Використання даного підходу в рамках системи, що розроблюється, зображено на рисунку 6.6.

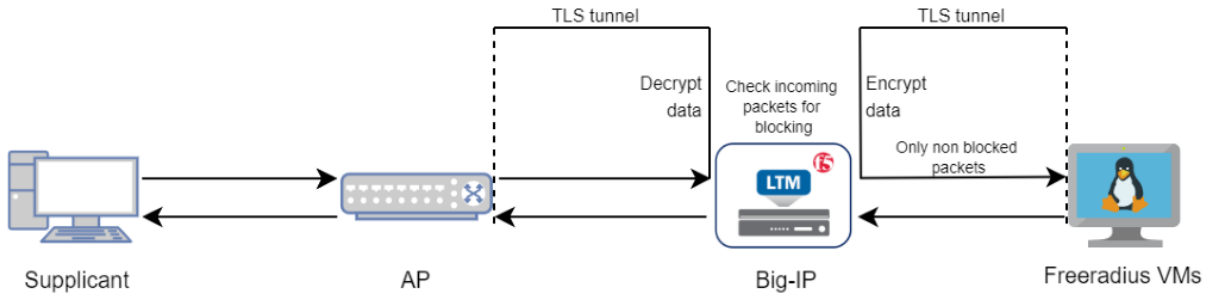

Рисунок 6.6 – Схема принципу роботи SSL Bridging

Найбільшим недоліком SSL Bridging є процес перезапису. Сервер отримує повноваження перевіряти та модифікувати дані клієнта, якщо буде в виявлено щось підозріле. Потім він повторно шифрує та надсилає лише той вміст, який він вважає безпечним. Якщо алгоритм фільтрації робить помилково позитивне рішення і блокує його, буде втрачено важливий і безпечний трафік.

При цьому навантаження від шифрування та дешифрування не забирається з сервера.

6.4 Проектування бази даних

В процесі проектування системи постало питання обрання бази даних: реляційної чи нереляційної. Було проведено порівняльний аналіз цих двох типів баз даних та на основі отриманих результатів буде спроектовано базу даних розроблюваної системи.

Реляційна база даних базується на реляційній моделі даних. [16] Такі дані зберігаються у таблицях: вони місять рядки, які представляють запис, та стовпчики, що зберігають певний тип інформації. Відносини між записами та таблицями встановлюються за допомогою первинного та вторинного ключів.

Переваги реляційних баз даних:

- можливість оброблювати складні запити;
- підтримують транзакції;
- підходять для аналізу даних;
- підтримують ACID (Atomicity, Consistency, Isolation, Durability) – набір властивостей, що забезпечують надійну роботу транзакцій бази даних, а саме атомарність, узгодженість, ізольованість та довговічність. [17]

Недоліками реляційних баз даних є наступні:

- не можуть зберігати складні чи дуже великі зображення, числа та мультимедійні файли;
- можуть дуже дорого обходитися в підтримці та створенні нових серверів.

Нереляційні бази даних надають механізм для збереження та отримання інформації, що не має чітко описаної структури. [18] Такі бази даних поділяються на наступні типи:

- стовпчикові – таблиці, що містять набір, зазвичай, стовпчиків та рядків, які зберігають та організовують дані;
- граф-орієнтовані – використовують графи для представлення та збереження даних;
- ключ-значення – використовують масив як модель даних, в яких дані зберігаються у вигляді колекції ключів-значень;

– документ-орієнтовані – є наполовину структурованими даними.

Перевагами нереляційних баз даних є:

– можливість зберігати записи в одній колекції, які мають різний набір полів та атрибутів;

– краща здатність до масштабування;

– можливість використовувати хмарні обчислення та сховища;

– швидка розробка, так як використання нереляційих баз даних потребують меншої підготовки, ніж реляційних;

До недоліків нереляційних баз даних можна віднести:

– система прив’язується до конкретної мови запитів обраної бази даних;

– відсутність транзакцій;

– обмежена можливість функцій мови запитів;

– при невдалій архітектурі схеми бази на початку в подальшому можна стикнутися з труднощами.

За результатами аналізу цих двох видів баз даних, було вирішено використовувати при розробці системи реляційну базу даних, адже дані в системі є структурованими, а при використанні систем не планується роботи з великою кількістю різноманітних даних. До того ж реляційні бази даних є більш уніфікованими, так як використовують мову запитів SQL, тому процес переходу між різними видами баз даних при потребі буде полегшеним. Схема розроблюваної бази даних зображено на схемі Г.

Таблиця devices зберігає дані про пристрої, що були заблоковані системою та вона містить наступні поля:

– device_id – первинний ключ;

– mac_address – інформація про MAC-адрес пристрою з запиту;

– username – ім’я користувача з запиту;

– port – номер порту, на який було відправлено запити;

– is_blocked – чи заблокований цей пристрій;

– should_auto_unblock – чи доступне пристрою авторозблокування.

Таблиця block_info зберігає дані про блокування, історію блокування в якомусь сенсі та містить вона наступне:

- block_info_id – первинний ключ;
- date – дата блокування.

Таблиця devices_block_info реалізує відношення один-до-багатьох, тобто в даній таблиці зберігається безпосередньо записи інформації про блокування кожного з пристроїв. Дана таблиця містить наступні поля:

- id – первинний ключ;
- block_info_id – ключ з таблиці devices_block_info;
- device_id – ключ з таблиці devices.

Таблиця users зберігає дані про користувача, загалом це інформація про одного користувача, а саме – системного адміністратора, що займається безпекою мережі, і містить наступні поля:

- user_id – первинний ключ;
- login – логін для входу:
- password_hash – хешований пароль користувача.

Таблиця block_settings зберігає параметри налаштувань блокування системою, а саме:

- id – первинний ключ;
- success_threshold – поріг для успішних автентифікацій;
- fail_threshold – поріг для невдалих автентифікацій;
- acct_threshold – поріг для запитів обліку;
- aggregate_period – період за який агрегуються дані;
- autounblock_enabled – чи дозволено авторозблокування;
- autounblock_period – період для авторозблокування;
- custom_udp_rule – користувацька функція для фільтру UDP пакетів;
- custom_tls_rule – користувацька функція для фільтру зашифрованих пакетів.

6.5 Проектування клієнтської та серверної частини

Для спілкування користувацького інтерфейсу, тобто клієнтського рівня, з серверною частиною системи вирішено використовувати архітектурний стиль REST API.

REST (REpresentational State Transfer) [19] – це архітектурний стиль, що задає правила для будування веб-сервісів. Системи, основані на даному підході, орієнтуються на швидке виконання, надійність та можливість росту шляхом повторного використання компонентів, що можуть бути керованими та поновленими без впливу на систему у цілому.

Перевагами даного підходу є наступні:

- підтримка різних форматів даних, таких як XML, HTML та JSON;
- легка структура, що зменшує навантаження на мережу та забезпечує оптимальне виконання;
- можливість керування, що забезпечує зменшення кількості даних, що надсилаються та отримуються.

- Але при цьому цей архітектурний стиль має і наступні недоліки:
- слабка типізація даних може спричинити помилки при спілкування
- між програмами, навіть між тими, що підтримують REST API;
- дані можуть бути або надлишковими або їх може бути недостатньо.

REST найкраще використовується для роботи з системами шляхом розділення інформації, що створюється та використовується, від технологій, що створюють та використовують її. Таким чином це дозволить досягти масштабованості, загальності та простоти системи.

Також для роботи з клієнтською частиною буде використано шаблон проектування MVC.

Так як статична сторінка на HTML не вміє реагувати на дії користувача, а для двосторонньої взаємодії потрібні динамічні веб-сторінки, то MVC є рішенням цієї проблеми.

MVC розшифровується як модель-представлення-контролер (від англ. Model-View-Controller). Це спосіб організації коду, який передбачає виділення блоків, що відповідають за вирішення різних завдань. Один блок відповідає за дані застосунка, інший відповідає за зовнішній вигляд, а третій контролює роботу самого застосунка.

Компонентами MVC є наступні:

– модель – цей компонент відповідає за дані, а також визначає структуру програми;

– представлення – це компонент, що відповідає за взаємодію з користувачем. Тобто код компонента визначає зовнішній вигляд програми і способи його використання;

– контролер - цей компонент відповідає за зв'язок між моделлю і представленням. Код даного компоненту визначає, як сайт реагує на дії користувача.

Перевагами використання такого шаблону проектування є те, що основна мета полягає в ізоляції частин коду, тим самим значно зменшуючи складність і тим самим збільшуючи легкість підтримки коду. Розробник клієнтської частини може змінювати візуальне представлення, модифікуючи HTML-код, не турбуючись про бізнес-логіку або спосіб отримання даних, тоді як розробник серверної частини може легко змінити бізнес-логіку, не впливаючи на презентацію. І якщо є потреба переключитися на іншу базу даних, можна просто поміняти рівень доступу до даних іншим, не змінюючи решту програми.

Серед недоліків такого підходу є те, що у програмі MVC застосунок дотримується чіткого процесу. Кожен запит направляється на дію, метод класу контролера. Там на основі параметрів отримується необхідний набір даних та відображається на клієнтській частині. Коли надсилається запит на отримання нової візуальної частини програми, йому потрібно знову пройти цей процес, який може бути повільним та вимагати великих ресурсів, якщо від користувацького інтерфейсу до серверу надходить багато запитів.

6.6 Алгоритм фільтрації пакетів

Алгоритм для захисту від DoS атак в основному базується на методі фільтрації пакетів, при цьому також використовуються характеристики порогового методу та метод виділення ознак.

В рамках порогового методу введено певний ліміт запитів за вказаний період часу (наприклад, 1 хвилину). А в рамках методу виділення певних ознак – використання даних про пристрій та сам запит, а саме ім’я користувача та MAC-адреса пристрою.

Під час аналізу, з отриманого пакету береться інформація про пристрій (ім’я користувача, MAC-адреса пристрою), тип пакету (автентифікація чи облік) та порт (у випадку, якщо сервер використовує велику кількість портів) на який надсилаються пакети і формується певний ключ, який перевіряється на наявність у базі даних мережевого екрану.

На етапі автентифікації в RADIUS-пакеті точно передається IP-адреса точки доступу, за допомогою якої пристрій намагається автентифікуватися, і не завжди IP-адреса самого пристрою, тому дана характеристика не використовується при аналізі.

Якщо даний ключ існує, то всі пакети з таким вмістом відхиляються, в протилежному випадку – пропускаються далі, при цьому агрегується кількість запитів певного ключа за вказаний період (наприклад 1 хвилину). Якщо це число перевищує попередньо визначений ліміт на кількість запитів автентифікації та обліку за одним ключем за період часу, то цей ключ додається до списку заблокованих. Алгоритм роботи даного методу зображено у додатку Д.

Таким чином, даний алгоритм включає в себе метод фільтрації пакетів та автоматизує фільтрацію за MAC-адресами, але при цьому є більш гнучким, бо використовуються і інші дані з пакету.

6.7 Опис взаємодії з іншими компонентами системи автентифікації

Описаний процес спілкування зображено у діаграмах послідовності та комунікації у додатках Е, Ж та И відповідно.

Розроблювана система знаходиться посередині процесу автентифікації.

Процес автентифікації користувача починається з пристрою. Він ініціює підключення до мережі шляхом запиту на точку доступу або інший сервер доступу до мережі.

Загалом, сервер мережевого доступу діє як шлюз між користувачем та ширшою мережею. Коли користувач намагається отримати доступ до мережі, NAS передає інформацію про автентифікацію (наприклад, ім'я користувача та пароль) між користувачем та сервером RADIUS. Цей процес називається сеансом автентифікації.

В кінці сеансу автентифікації сервер доручає NAS відхилити користувача та заборонити доступ до мережі або прийняти користувача та надати доступ до мережі. Після того, як користувач отримав доступ до мережі, обмеження безпеки (визначені сервером RADIUS) застосовуються NAS, який виконує роль маршрутизатора шлюзу та брандмауера для цього користувача. Сервер RADIUS отримує короткий опис діяльності користувача від NAS. Цей підсумок включає такі дані, як інформація про ідентифікацію сеансу, загальний час роботи в мережі та загальний трафік до користувача та від нього.

Користувацький трафік не проходить через сервер RADIUS – сервер RADIUS має доступ до інформації про користувача лише через зведення NAS.

В рамках розроблюваної системи при обміні пакетами між сервером доступу до мережі та сервером автентифікації знаходить мережевий екран, що фільтрує отримані пакети. Коли пакет надходить до цього шару, то тут відбувається отримання даних з отриманого пакету або у випадку RadSec – попереднє дешифрування даних цього пакету. Якщо дані з даного пакета не знаходяться у переліку заблокованих, то пакет пропускається далі до серверу автентифікації. Якщо ні, то блокується.

Коли сервер автентифікації отримує пакет та встановлює спілкування з сервером доступу до мережі починається процес автентифікації. Під час цього

процесу йду запит до бази даних, та вирішується чи може даний користувач бути під’єднаним до мережі. Також ці дані надсилаються до серверу захисту, де в процесу аналізується чи були певні порушення політик системи та чи є трафік з даного пристрою сумнівним.

Якщо політики було порушено, то дані про пристрій та користувача надсилаються до мережевого екрану, де в подальшому, як було зазначено, пакети з цими даними будуть відхилятися.

6.8 Висновки розділу

У даному розділі було описано архітектуру системи та алгоритм фільтрації пакетів.

Було зазначено, що у роботі мережевого екрану буде використовуватися принцип SSL Bridging, що дозволяє розшифровувати зашифрований трафік у випадку використання RadSec.

Для спілкування клієнтської та серверної частин було обрано архітектурний стиль REST, так як він підтримує декілька форматів даних, є легким, тобто не навантажує використання мережі шляхом отримання лише потрібних даних, та дозволяє кешувати отримані дані, що дозволяє зменшити кількість операцій з даними. Також буде використано шаблон проектування MVC.

Також було зазначено які технології будуть використовуватися при розробці з їх перевагами та недоліками, а саме платформа .NET, мова програмування C++, сервер автентифікації FreeRADIUS та інструменти хмарної платформи Microsoft Azure.

В якості типу бази даних було вирішено використовувати реляційний тип, адже в розроблюваній системі дані є доволі структурованими та не планується робота з великою кількістю різноманітних даних. Обрана база даних для реалізації системи – MySQL.

7 РЕАЛІЗАЦІЯ СИСТЕМИ

7.1 Сервер системи захисту

Серверна частина реалізована на мові C# та за допомогою технологій хмарної платформи Azure та ASP.NET.

Реалізація серверу складається з двох частин: логіки обробки інформації з пакетів для аналізу та робота з даними для веб-інтерфейсу для зручного менеджменту.

Далі описану реалізацію у вигляді структурної схеми да діаграми розгортання зображено у додатку К.

7.1.1 Обробка даних з пакетів

Для аналізу кількості запитів за певний проміжок часу було використано технології Event Hub, Service Bus та Stream Analytics.

Azure Event Hubs [20] – це велика платформа для потокового передавання даних та служба передачі подій. Він може приймати і обробляти мільйони подій в секунду. Дані, що надсилаються до Event Hub, можуть бути перетворені та збережені за допомогою будь-якого постачальника аналітики в режимі реального часу або адаптерів зберігання даних.

Концентратори подій представляють «вхідні двері» для конвеєра подій, який часто називають поглиначом подій в архітекторах рішень. Організатор подій – це компонент або послуга, яка розміщується між створювачами та споживачами подій, щоб відокремити виробництво потоку подій від споживання цих подій. Event Hub забезпечує уніфіковану потокову платформу з буфером збереження часу, відокремлюючи виробників подій від їх споживачів подій.

Microsoft Azure Service Bus [21] – це повністю керований посередник корпоративних повідомлень з чергами повідомлень та темами для загальнодоступної підписок. Сервісна шина використовується для роз'єднання програм і служб один з одним, забезпечуючи такі переваги, як робота з вирівнювання навантаження у

конкуруючих робітників; безпечна маршрутизація, передача даних та контроль через межі служб та застосунків; координація транзакційних робіт, що вимагає високого ступеня надійності.

Тема розроблюваної системи в сервісній шині в інтерфейсі порталу Azure зображено на рисунку 7.1

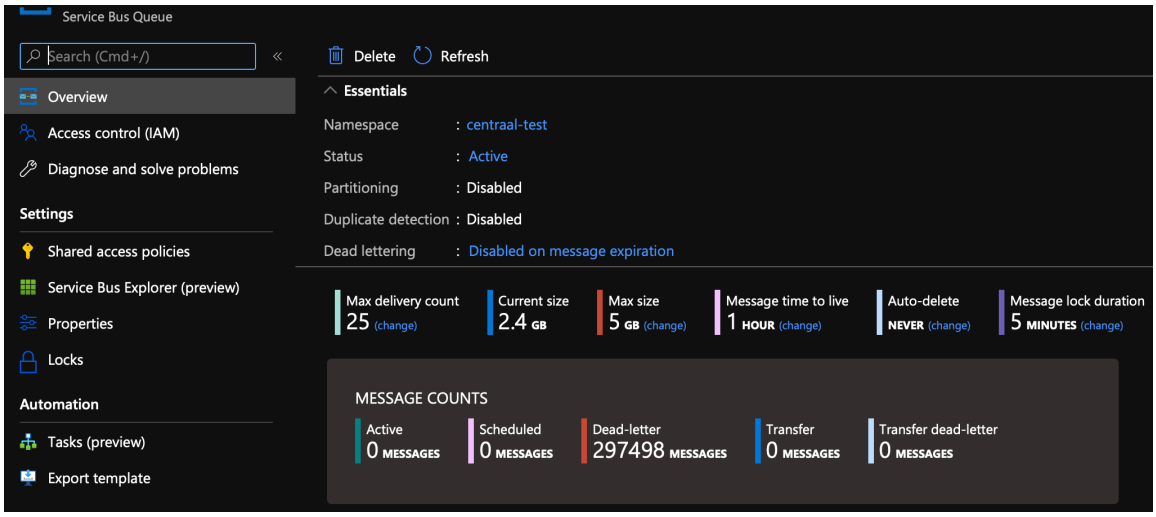

Рисунок 7.1 – Тема для роботи з результатами Stream Analytics

Azure Stream Analytics [22] – це аналітика в реальному часі та складний механізм обробки подій, який призначений для аналізу та обробки великих обсягів швидкого потокового передавання даних з кількох джерел одночасно. Шаблони та взаємозв'язки можна визначити в інформації, витягнутій з багатьох джерел вхідних даних, включаючи пристрої, датчики, стрічки соціальних мереж та застосунки. Ці шаблони можна використовувати для активації дій та ініціювання робочих процесів, таких як створення попереджень, подача інформації до інструменту звітування або зберігання трансформованих даних для подальшого використання. Крім того, Stream Analytics доступна в середовищі виконання Azure IoT Edge, що дозволяє обробляти дані на пристроях IoT.

Завдання Azure Stream Analytics складається з введення, запиту та результату. Stream Analytics передає дані з концентраторів подій Azure, концентратора IoT Azure або сховища BLOB-об'єктів Azure. Запит, який базується на мові запитів SQL, можна використовувати для легкого фільтрування, сортування, агрегування та приєднання

потокових даних протягом певного періоду часу. Також можна розширити цю мову SQL за допомогою визначених користувачем функцій JavaScript та C# (UDF). Можна легко налаштувати параметри упорядкування подій та тривалість часових вікон під час виконання операцій агрегування за допомогою простих мовних конструкцій та/або конфігурацій.

У випадку розроблюваної системи було описано роботу Stream Analytics за допомогою функцій з використанням мови JavaScript. Приклад такої функції наведено у лістингу А.4.

Для агрегації даних було використано запити SQL, приклад якого наведено у лістингу А.5.

Існує декілька підходів до агрегації даних в Stream Analytics за певний період часу:

– Tumbling вікно – функції цього вікна використовуються для сегментації потоку даних на різні часові сегменти та виконання функції над ними. Ключовими відмінами даного вікна є те, що вони повторюються, не перекриваються, і подія не може належати більше ніж одному вікну.

– Hopping вікно – функції вікна «стрибають» вперед у часі на фіксований період. Їх можна сприймати як перекидні вікна, які можуть перекриватися та випускатися частіше, ніж розмір вікна. Події можуть належати більше ніж одному набору результату вікна.

– Sliding вікно – на відміну від tumbling або hopping вікон, ці вікна видають події лише для моментів часу, коли вміст вікна фактично змінюється. Іншими словами, коли подія входить або виходить з вікна. Отже, у кожному вікні є принаймні одна подія. Подібно до hopping вікон, події можуть належати до кількох tumbling вікон.

– Session вікно – функції групують події, що надходять у подібний час, фільтруючи періоди часу, коли немає даних. Він має три основні параметри: час очікування, максимальна тривалість та необов’язковий ключ розділення.

– Snapshot вікно – дане вікно групує події, що мають однакову позначку часу. На відміну від інших типів вікон, для яких потрібна певна функція вікна

(наприклад, SessionWindow (), можна застосовувати snapshot вікно, додавши System.Timestamp () до речення GROUP BY.

В результаті порівняння, для підрахунку за певний проміжок часу було обрано підхід tumbling вікна, так як воно є серією фіксованих розмірів, які не перекриваються та є суміжними інтервалів часу, що і потрібно в рамках реалізації системи.

Загальний принцип роботи зв'язки Event Hub, Service Bus та Stream Analytics зображено на рисунку 7.2.

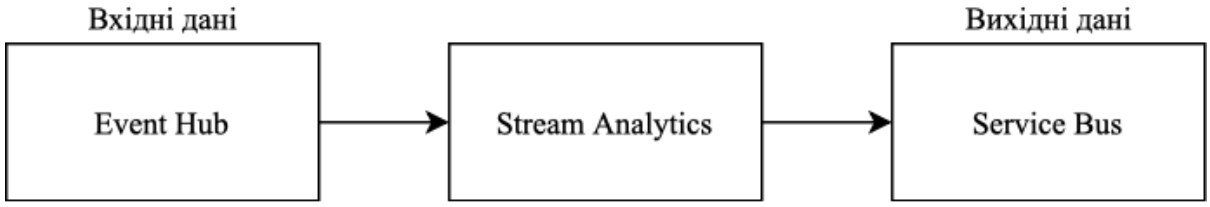

Рисунок 7.2 – Схема роботи Event Hub, Service Bus та Stream Analytics

Таким чином Stream Analytics отримує дані з Event Hub, далі оброблює їх і надсилає їх далі до Service Bus, звідки в подальшому сервер отримує дані для аналізу.

На стороні сервера обробка даних з Stream Analytics виконується фоновим процесом, який отримує регулярно дані агреговані за допомогою раніше зазначених віконних проміжків з вже зазначеного Service Bus. При отриманні не нульових даних формується ключ з даних про пристрій та надсилається до мережевого екрану.

7.1.2 Веб-застосунок для менеджменту системи

Також в рамках даного серверу можлива робота з даними про заблоковані пристрої, їх розблокування та налаштування користувацьких параметрів блокування за допомогою ASP.NET, а саме використовуючи MVC патерн.

Так, є контролер RadiusDosController, що відповідає за отримання даних про заблоковані пристрої, можливість їх розблокування. Серед його методів є наступні:

– public async Task<JsonResult> GetBlockedDevices(int pageNumber, int pageSize);

– public async Task<JsonResult> UnblockDevices(IEnumerable<BlockedDeviceInfoModel> devices).

– Ще один контролер RadiusDosSettingsController відповідає за управління даними про параметри блокування, тобто, наприклад, редагування значення порогу або інтервал агрегації запитів. Його методами є наступні:

– public async Task<JsonResult> GetDosSettings();

– public async Task<JsonResult> UpdateDosSettings (DosSettingsViewModel).

Дані про заблоковані пристрої та налаштування процесу блокування зберігаються у реляційній базі даних MySql. Доступ до даних реалізовано за допомогою шару доступу до даних.

Було створено клас BlockedDeviceInfoDataAccess та інтерфейс IBlockedDeviceInfoDataAccess, що відповідають за отримання, видалення та редагування даних про заблоковані пристрої.

Контрактом передбачені наступні операції з заблокованими сутностями:

– Task<BlockedDeviceInfo> GetAsync(string blockedDeviceId);

– Common.Utils.Iterator.IAsyncEnumerator<BlockedDeviceInfo> GetListAsync();

– Task<PagedResult<BlockedDeviceInfo>> GetPageAsync(string orgId, int pageNumber, int pageSize);

– Task<IEnumerable<BlockedDeviceInfo>> GetUpdatedAsync (DateTime startDate);

– Task<IEnumerable<BlockedDeviceInfo>> GetDeletedAsync (DateTime startDate);

– Task<IEnumerable<BlockedDeviceInfo>> GetAsync (IEnumerable<string> macAddresses);

– Task<IEnumerable<string>> GetExistedAsync (IEnumerable<string> ids);

– Task<IEnumerable<BlockedDeviceInfo>> GetByIdsAsync (<string> ids);

– Task<IEnumerable<BlockedDeviceInfo>> UpsertBatchAsync (IEnumerable<BlockedDeviceInfo> blockedDeviceInfos);

– Task BatchDeleteAsync (IEnumerable<string> ids);

– Task DeleteAllAsync().

Для роботи з налаштуваннями блокування створено інтерфейс IDevicesBlockingSettingsDataAccess та його реалізація DevicesBlockingSettingsDataAccess.

Даний контракт передбачає наступні операції:

– Task<DevicesBlockingSettings> GetAsync();

– Task<bool> DeleteAsync();

– Task<DevicesBlockingSettings> UpsertAsync (DevicesBlockingSettings settings).

Проміжний шар між контролером та класом BlockedDeviceInfoDataAccess, що реалізує головні функції для отримання та роботи з заблокованими пристроями реалізує наступні методи:

– public async Task UnblockDdosDevicesAsync(UnblockDevicesRequest request)

– public async Task<GetBlockedDevicesPagedResponse> GetDdosBlockedDevicesAsync(int pageNumber, int pageSize)

– public async Task<GetBlockedDevicesResponse> GetDdosBlockedDevicesByMacAddressesAsync(IEnumerable<string> macAddresses)

Для реалізації функціоналу блокування та розблокування на мережевому екрані через сервер системи захисту створено інтерфейс IDevicesBlocker та його реалізація, що передбачає наступний контракт:

– Task<IEnumerable<BlockedDeviceInfo>> BlockDevicesAsync (IEnumerable<BlockedDeviceInfo> devices);

– Task UnblockDevicesAsync (IEnumerable<BlockedDeviceInfo> devices);

У реалізації цих методів використовуються API запити до мережевого екрану з метою додавання чи видалення з нього інформації про заблокований пристрій.

7.2 RADIUS-сервер

В даному пункті описується частина, що зображена на рисунку 7.3, яка є деталлю діаграми В.

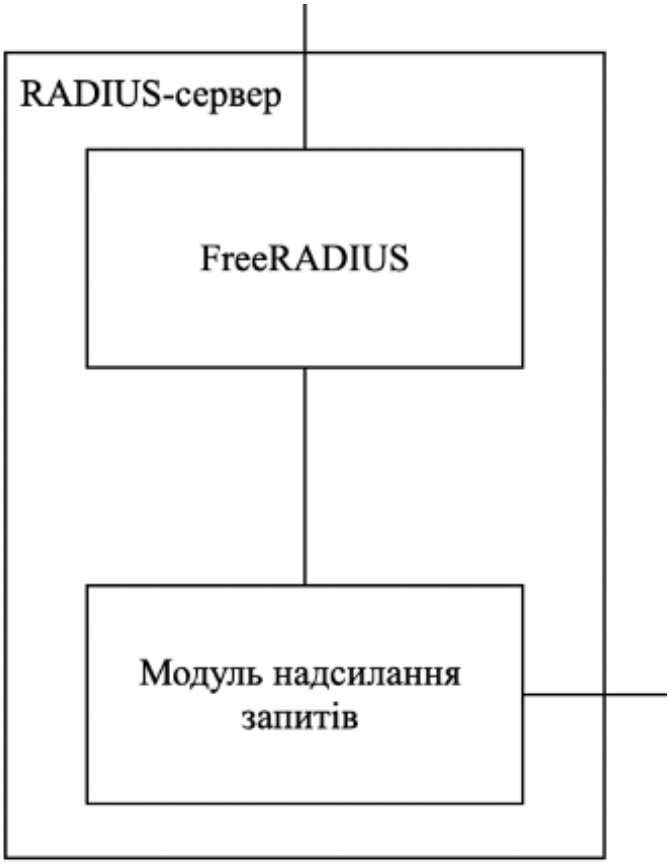

Рисунок 7.3 – RADIUS частина діаграми компонентів

Для RADIUS-серверу було обрано FreeRADIUS. Він є доволі легким в налаштуванні, але для того, щоб задовільнити потреби розроблюваної системи було розширено його функціонал та додано власні налаштування.

Також для створення можливості направляти події про автентифікацію чи облік до Azure Event Hub, тим самим надсилати дані для подальшої обробки в Azure Stream Analytics, було створено модуль надсилання запитів, що є фоновою програмою-процесом, яка отримує дані з FreeRADIUS та направляє її до Event Hub.

Алгоритм роботи даної логіки наступний: після завершення обробки RADIUS-сервером події (автентифікації чи обліку) надсилається інформація до фонової

програми-процесу. В рамках її роботи з отриманого повідомлення отримується інформація про подію та формується повідомлення до Event Hub.

7.2.1 Розширення функціоналу FreeRADIUS

По-перше необхідно додати налаштування для спілкування серверу та допоміжної програми. У FreeRADIUS конфігурація створена за допомогою мови unlang, що є простою мовою обробки конфігураційних файлів. Мета мови – дозволити писати прості правила з мінімальними зусиллями. Потім ці політики застосовуються під час обробки запиту.

Синтаксис unlang базується на зарезервованих словах, які диктують, як слід інтерпретувати текст, що йде далі. Синтаксис кожного слова орієнтований на рядки. Іноді дозволяються підрозділи, як і в інших частинах radiusd.conf. Якщо рядки занадто довгі, символ зворотної косої риски можна використовувати для зв’язку декількох рядків.

Для досягнення потрібної мети, було розширено головний конфігураційний файл серверу – radiusd.conf, що відповідає за базові налаштування роботи FreeRADIUS.

Коли сервер запускається, він зчитує цей файл і кешує його. Дані аналізуються для встановлення значень змінних або для визначення іншої конфігурації, наприклад модулів. Після завантаження конфігурації сервер отримує та обробляє пакети. Коли сервер працює в режимі налагодження, використовувана конфігурація друкується у поточному вікні терміналу. Ця інформація включає детальну інформацію про файли, які зчитуються, модулі, які завантажуються, а також імена та значення будь-яких використовуваних налаштувань.

Базові налаштування даного файлу зображено на рисунку 7.4

```
prefix = /usr
exec_prefix = /usr
sysconfdir = /etc
localstatedir = /var
sbindir = ${exec_prefix}/sbin
logdir = ${localstatedir}/log/freeradius
raddbdir = ${prefix}/local/etc/raddb
radacctdir = ${logdir}/radacct

name = freeradius

confdir = ${raddbdir}
modconfdir = ${confdir}/mods-config
certdir = ${confdir}/certs
cadir   = ${confdir}/certs
run_dir = ${localstatedir}/run/${name}

db_dir = ${raddbdir}
libdir = ${prefix}/local/lib
pidfile = ${run_dir}/${name}.pid

correct_escapes = true

max_request_time = 30
cleanup_delay = 5
max_requests = 1024

hostname_lookups = no

log {
        destination = ""
        syslog_facility = local1
        stripped_names = yes
        auth = yes
        auth_badpass = no
        auth_goodpass = no
#       msg_goodpass = ""
#       msg_badpass = ""
}
```

Рисунок 7.4 – Базові налаштування radiusd.conf

До нього було додано нову секцію external_communication, що містить посилання на фонову програму. Було обрано саме секцію, а не просто рядок з налаштуванням для, того щоб, при потребі розширити функціонал роботи з фоновою програмою, всі налаштування пов’язані з цим належали до одної секції.

Щоб додана конфігурація працювала необхідно також додати інформацію про нею у mainconf.c, що оброблює конфігураційний файл. Для цього для масиву server_config було додано рядок {" external_communication", FR_CONF_POINTER(PW_TYPE_SUBSECTION, NULL), (void const *) external_communication}.

Даний рядок зазначає, що назва «external_communication» в конфігураційному файлі є секцією, тобто типом PW_TYPE_SUBSECTION, і має базову конфігурацію у вигляді структури external_communication, якщо в самому файлі дана секція не була зазначена.

Сама структура external_communication повторює вже зазначено раніше, а саме – вона має одне поле logging_url типу char const, що містить адресу фонової програми для відправки логів.

Для відправки логів було додано метод, що надсилає потрібні дані до серверу фонової програми за допомоги інструменту curl.

В рамках розроблюваної системи нас цікавить дві події: автентифікація та облік. Для моніторингу автентифікації було додано в місцях, де є результат автентифікації такий як Access-Accept або Access-Reject файлу auth.c надсилання логу.

Для моніторингу подій обліку, було створено власну реалізацію роботи з обліком вигляді модуля, що дозволяє розширити базовий функціонал базового модуля обліку FreeRADIUS. Для підтримки створеної реалізації обліку необхідно було додати в файлі конфігурації, що відповідає за роботу з обліком додати секцію custom_event event_accounting, а потім зазначити у базових налаштуваннях серверу, що облік буде оброблюватися за допомогою зазначеного event_accounting, шляхом додавання рядка accounting {event_accounting }.

Розроблений модуль використовує базову логіку модуля обліку, але при обробці події він надсилає інформацію про запит до серверу фонової програми за таким самим принципом, що і при автентифікації.

Частину реалізації даного модуля зображено на рисунку 7.5.

```
if (inst->type == ACCT_TYPE) {
    /* accounting subtype should be only start or stop */
    dstr subtype_val = get_acct_subtype(request);
    char *subtype_val_str = dstr_to_cstr(&subtype_val);
    int subtype_result = 0;

    if (dstr_size(&subtype_val) == 0) {
        ERROR("rlm_event: FAILED. Event type %s should contains 'Acct-Status-Type' attribute, on port %s", type_map[inst->type], n_str(request->clien
        subtype_result = -1;
    } else if (strcmp(subtype_val_str, START_ACCT_SUBTYPE) == 0) {
        subtype = SUBTYPE_START;

    } else if (strcmp(subtype_val_str, STOP_ACCT_SUBTYPE) == 0) {
        subtype = SUBTYPE_STOP;
    } else {
        INFO("rlm_event: FAILED. Event type %s contains wrong 'Acct-Status-Type' attribute '%s', should be 'Start' or 'Stop', on port %s",
                type_map[inst->type], subtype_val_str, n_str(request->client_shortname));
        subtype_result = -1;
    }

    dstr_destroy(&subtype_val);
    if (subtype_result != 0) return RLM_MODULE_OK;
}
```

Рисунок 7.5 – Частина реалізації модуля обробки подій

7.2.2 Модуль надсилання запитів

Даний модуль є фоновою програмою-процесом, що розроблена з використанням мови C++. Вона працює на фоні як сервер, що отримує з RADIUS-серверу інформацію про запити, потім оброблює та форматує отримані дані та в кінці надсилає їх до Event Hub.

Вся реалізація базується на фонових процесах, для чого було створено головний клас, що за це відповідає, а саме DaemonService.

На його основі було створено базовий сервіс, що має такі публічні функції:

- int Run();
- int Stop();
- string Name();
- DaemonService* Create().

Серед приватних функцій є наступні:

- void SetupEndpoints() – конфігурація сервісів;
- WebServiceResponse ProcessAccounting(string& data, UrlParams& urlParams) – обробка обліку;
- WebServiceResponse ProcessLogs(string& data, UrlParams& urlParams) – обробка логів.

А також було створено сервіс RadiusEventsSenderService, що реалізує логіку відправки до Event Hub. Головний його метод, тобто метод надсилання інформації до з Event Hub ображена на рисунку 7.6.

```
void RadiusEventsSenderService::SendData(DataPublisherSettings& eh_settings) {
    try {
        // if eventhub data is not retrieved from BE
        if (eh_settings.radius_events_msg_type == nullptr || eh_settings.radius_events_msg_type->empty()) {
            logs::put(
                    format: "RadiusEventsSenderService::SendData failed to send network packets, eh_settings.radius_events_msg_type is null or empty");
            return;
        }
        // do all work
        auto msg_type = *eh_settings.radius_events_msg_type.get();
        data_publisher::send_dumps(msg_type, *eh_settings.eventhub_settings.get());
    } catch (exception& ex) {
        logs::error(PROCESS_RADIUS_EVENTS_ERROR_CODE, this->Name().c_str(), PROCESS_RADIUS_EVENTS_ERROR_MSG, ex.what());
    }
}
```

Рисунок 7.6 – Метод сервісу для надсилання інформації до Event Hub

Структура класів зазначених сервісів, зображено на рисунку 7.7.

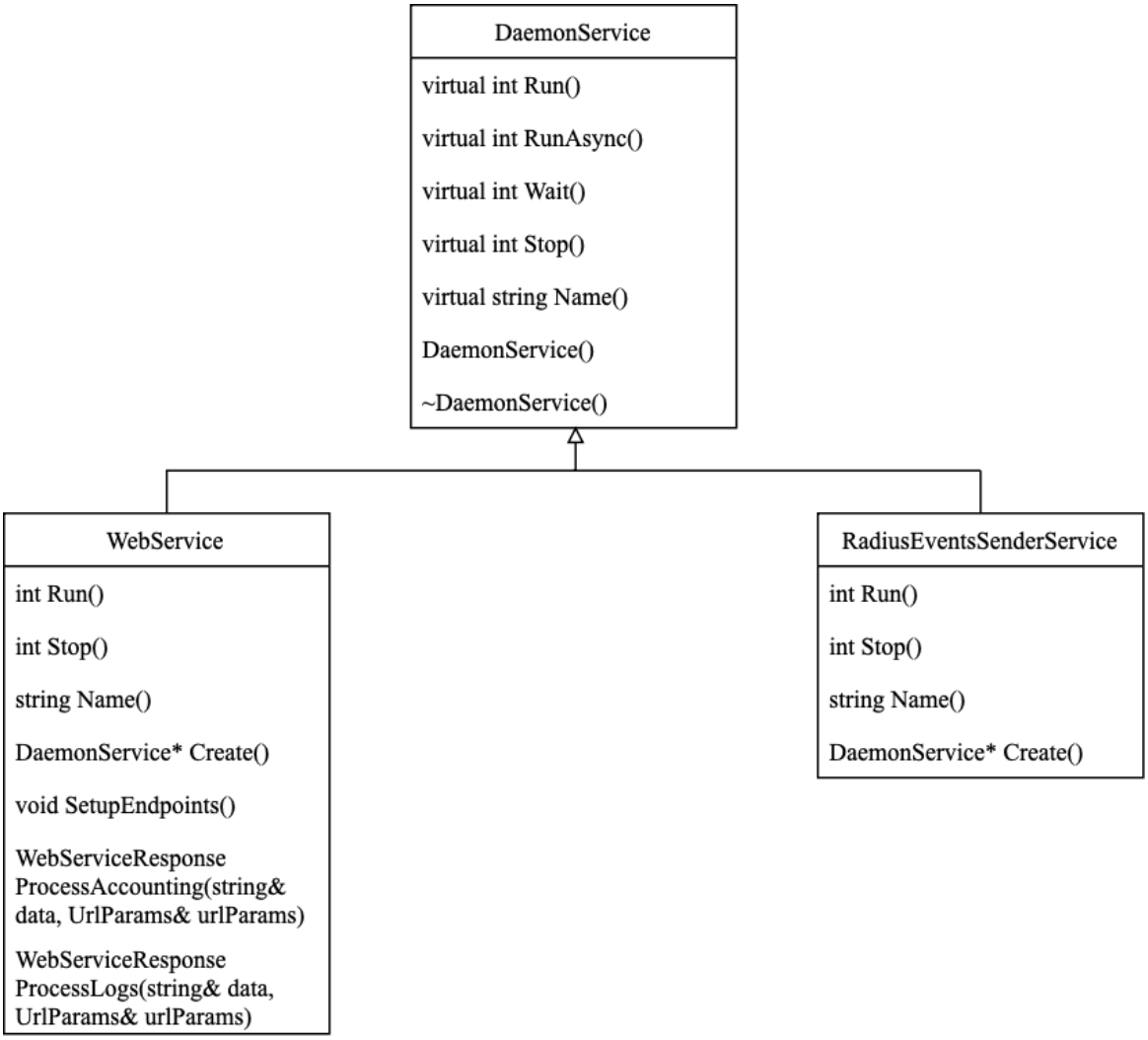

Рисунок 7.7 – Структура класів сервісів програми-процесу

Для роботи з Event Hub було створено допоміжний простір імен data_publisher, що зазначає контракти потрібних функції, серед яких є наступні:

– void dump(string msg, AppSettings& settings, string& partition, bool compress = true) – підготовлює дані для подальшого надсилання;

– int send_dumps(string& msg_type, EvenHubSettings& eh_settings) – безпосередньо надсилає інформацію до Event Hub;

– void invalidate_creds() – за потреби поновлює дані для доступу до Event Hub.

А на рисунку 7.8 зображено метод надсилання інформації про події з RADIUS-серверу, що викликається під час роботи сервісу.

```
int RadiusEventProcessor::DumpRadiusEventRequest(RadiusEventRequest request {
    // should dump only accounting requests
    if (request.event_type ≠ RadiusEventType::Accounting) {
        return HTTP_CODE_200;
    }

    auto eh_settings = ctx::settings.eh_settings();
    // if eventhub data is not retrieved from BE
    if (eh_settings == nullptr ||
        eh_settings→radius_events_msg_type == nullptr ||
        eh_settings→radius_events_msg_type→empty()) {
        logs::put("DumpRadiusEventRequest failed save dump of network packet, eh_settings.radius_events_msg_type is null or empty");
        return HTTP_CODE_500;
    }

    // dump data, will be processed inside RadiusEventsSenderService service
    auto msg_type = *eh_settings→radius_events_msg_type.get();
    auto settings = ctx::settings.settings();
    data_publisher::dump(JSON_TO_STR(request), settings, msg_type);
    return HTTP_CODE_200;
}
```

Рисунок 7.8 – Реалізація надсилання даних про події з RADIUS-серверу

7.3 Мережевий екран

Для реалізації мережевого екрану було обрано технологію BIG-IP[23], а саме одну з можливостей LTM (Local Traffic Managment). За допомогою даної технології можна керувати тим як саме трафік буде оброблятися перед його обробкою сервером RADIUS.

Також даний застосунок надає можливість обробляти трафік, що проходить через нього, за допомогою технології iRules [24]. iRule – це потужна та гнучка функція в системі управління локальним трафіком BIG-IP, яку можна використовувати для управління мережевим трафіком. iRules не тільки дозволяє вибирати пули на основі даних заголовків, але також дозволяє спрямовувати трафік шляхом пошуку в будь-якому типі даних вмісту, який ви визначаєте. Таким чином, функція iRules значно покращує здатність налаштовувати перемикання вмісту відповідно потреб.

Для реалізації потреб системи, використано можливості iRules для отримання даних з RADIUS-пакетів. Приклад функції наведено у лістингу А.1, а інтерфейс BIG-IP для роботи з ними на рисунку 7.9.

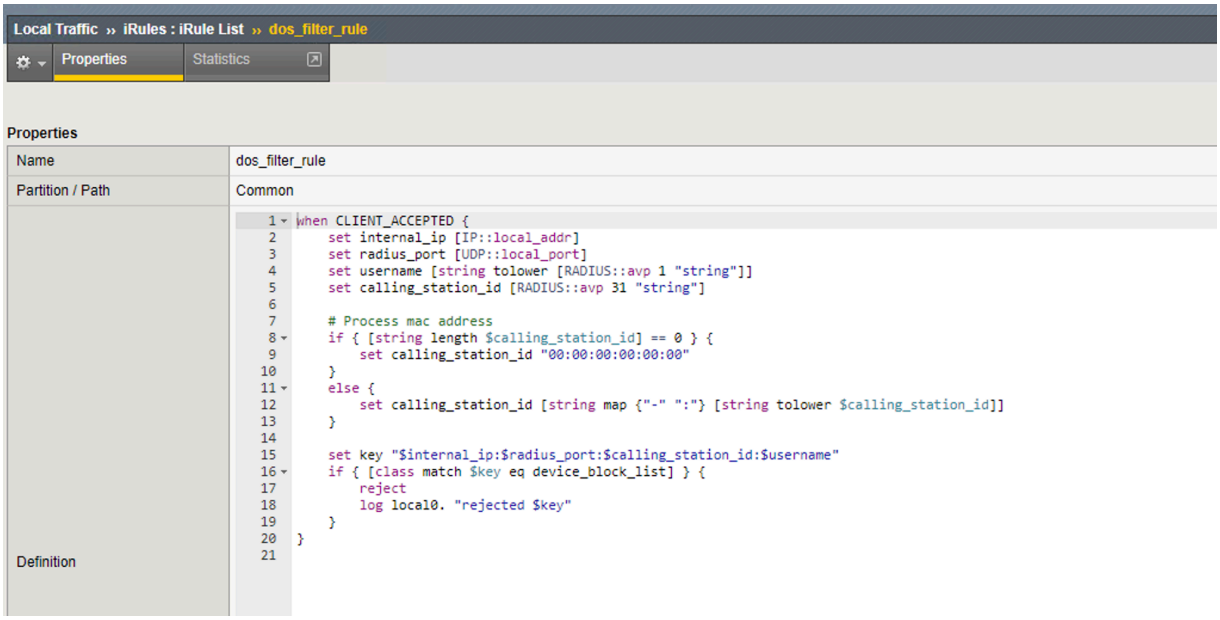

Рисунок 7.9 – Інтерфейс роботи з iRules

Для збереження ключів, за якими йде перевірка чи повинен бути заблокований певний пакет, використовується інша можливість системи BIG-IP – Data-Group [25] (далі – група даних). Це є особливим типом структури пам'яті в iRules. Фактично, це – список у форматі пари ключ-значення, що складається з IP-адрес/підмереж, рядків або цілих чисел. Можна мати лише список ключів або список відповідних ключів та значень у цій унікальній структурі. Дана структура є унікальною та особливою у декількох сеансах, але найбільш очевидним серед них є те, що він насправді постійно зберігається як частина конфігурації, а не лише як частина iRule або в пам'яті. Його можна зробити або вбудованим у файл big-ip.conf, або як окремий файл (відомий як зовнішня група даних). Обидва функціонують фактично однаково. Оскільки вони зберігаються в коробці, а не лише в пам'яті, групи даних попередньо заповнюються даними перед тим, як iRule коли-небудь виконуватиметься. Це робить їх ідеальними для збережених списків будь-якого роду. Змінити групу даних також просто завдяки прямому доступу через CLI або GUI. Можна додавати, змінювати або видаляти

записи в групі даних, ніколи не торкаючись iRule, що посилаються на неї, що робить його ідеальним для зберігання конфігураційних частин або статичної інформації, яку, можливо, потрібно час від часу оновлювати. Поки форматування даних залишається правильним під час редагування, немає реальних шансів зламати iRule, змінивши групу даних, оскільки сам код не торкається.

З цією метою існує два способи зберігання груп даних, внутрішній або зовнішній. Для внутрішніх груп даних набір даних зберігається у файлі bigip.conf. Для зовнішніх груп даних вони зберігаються у власному файлі та посилаються на них із об’єкта групи даних. Дуже великі набори даних слід зберігати у зовнішніх групах даних. Єдиним обмежувальним фактором щодо груп даних, що є об'єктом конфігурації, є те, що iRules не можуть впливати безпосередньо на об'єкти конфігурації.

До того ж, записи в групах даних зберігаються індексованими та хешований, а отже доступ до них є швидким, що якраз потрібно для реалізації системи.

Для реалізації системи було обрано використання внутрішніх груп даних, адже хоч потенційно для атак можуть використовуватися велика кількість пристроїв, тим не менш навряд чи їх кількість буде перевищувати 10000. Приклад внутрішньої групи даних у інтерфейсі BIG-IP зображено на рисунку 7.10.

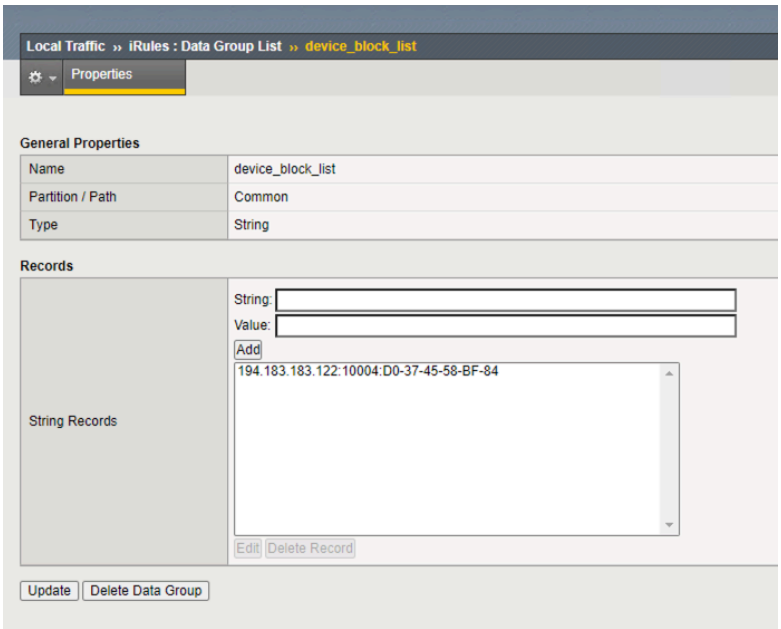

Рисунок 7.10 – Внутрішня група даних на BIG-IP

Важливо також зазначити, що за допомогою iRule можна реалізувати SSL Bridging для RadSec. Тобто під час виконання функції iRule, пакет буде розшифровано за допомогою попередньо доданого сертифікату, далі буде реалізовано аналіз пакету як і при роботі зі звичайним UDP протоколом, після аналізу та перед виходом з даної функції пакет знову буде зашифровано.

Приклад правила для роботи з зашифрованим трафіком наведено у лістингу А.2.

Для коректної роботи з розшифровуванням даних, у налаштуваннях BIG-IP було додано сертифікат RADIUS-серверу як зазначено на рисунку 7.11.

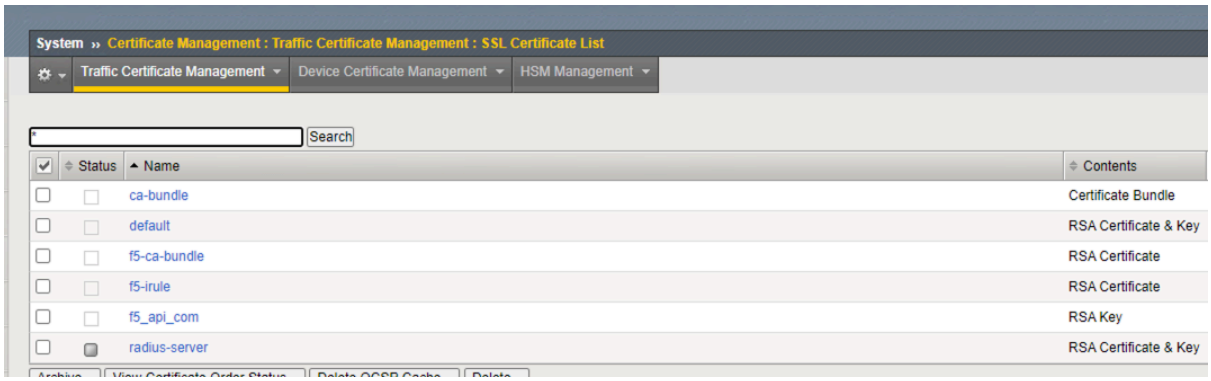

Рисунок 7.11 – Доданий сертифікат на BIG-IP

Для того, щоб система BIG-IP обробляла трафік застосунку через SSL, необхідно налаштувати систему на виконання SSL handshake, яке зазвичай виконують сервери призначення. Найпоширенішим способом налаштування BIG-IP для цього є створення клієнтського SSL-профілю, що дозволяє системі BIG-IP дешифрувати клієнтські запити перед надсиланням їх на сервер та шифрувати відповіді сервера перед тим, як відправляти їх назад на клієнт.

Тобто було створено SSL профіль, що відповідає безпосередньо за процес SSL Bridging, його зображено на рисунку 7.12.

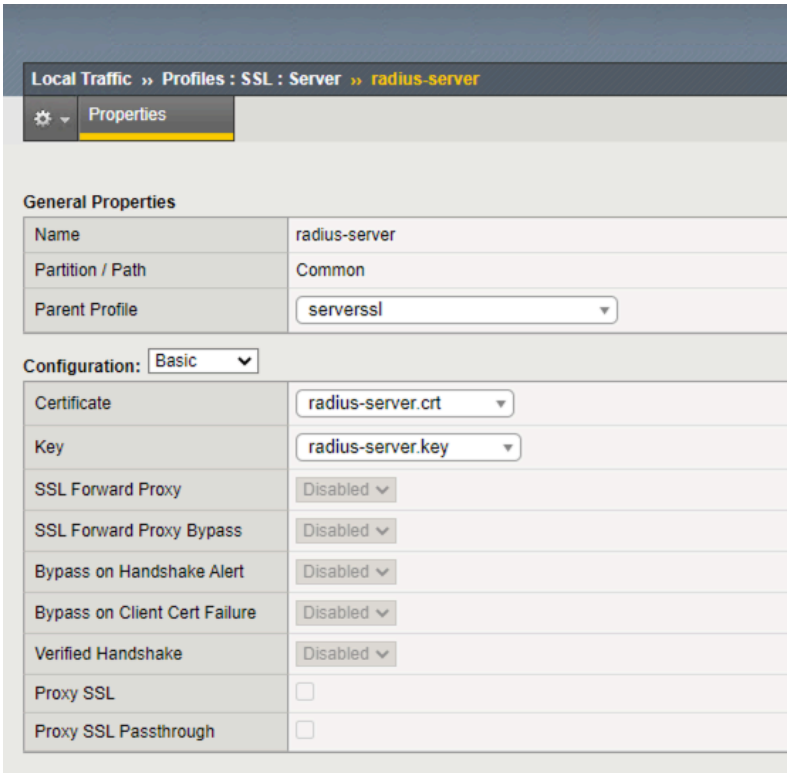

Рисунок 7.12 – Доданий профіль для SSL Bridging на BIG-IP

Для того, щоб дана конфігурація була використана, тобто принцип роботи SSL Bridging був задіяним, створений профіль необхідно додати до віртуального серверу, що керує перенаправленням трафіку. Налаштування віртуального серверу зображено на рисунку 7.13.

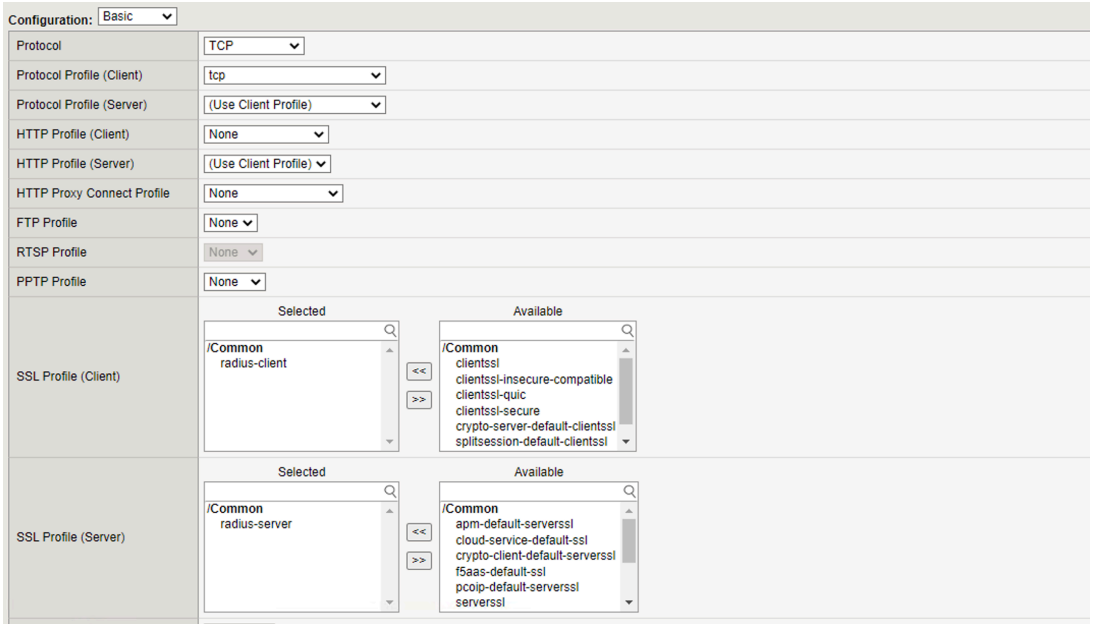

Рисунок 7.13 – Налаштування віртуального серверу

На рисунку 7.14 зображено перелік системних та доданих iRules правил.

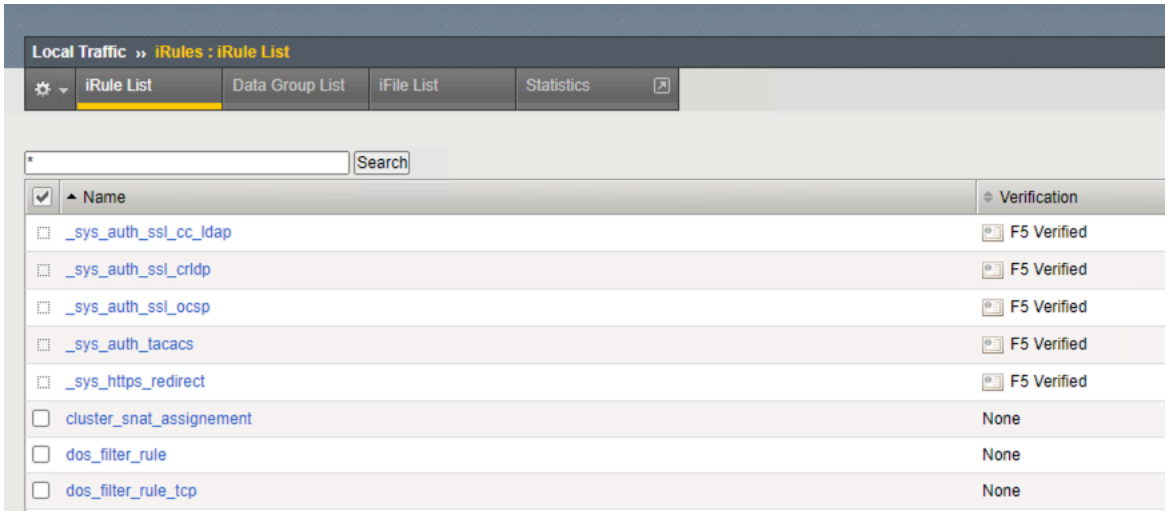

Рисунок 7.14 – Перелік правил iRules на BIG-IP

Робота з BIG-IP може бути також проведена за допомогою API, що дозволяє віддалено з серверу надавати команди для роботи з групами даних, а саме видаляти чи додавати записи в них. А також доступна можливість редагування, створення та видалення функцій iRule, що можуть знадобитися під час користувацького налаштування правил блокування. Приклад методу, що працюють з BIG-IP за допомогою API наведено у лістингу А.3.

7.4 Клієнтська частина

Рівень презентації, тобто інтерфейс, з яким працює клієнт розроблено у форматі одно сторінкового застосунку. Використання такого підходу забезпечує швидкість, так як ресурси, що використовуються на сторінці, такі як HTML та CSS, завантажуються лише один раз під час її завантаження.

В такому випадку дані або отримуються з сервера, або надсилаються на нього. Також для зменшення кількості запитів можна кешувати данні та використовувати їх для роботи, у випадку відсутності Інтернет з’єднання.

Дана частина виконана з використанням фреймворку React.js. За його допомогою було використано компонентний підхід до розробки клієнтської частини, тим самим збільшивши кількість коду, який можна використовувати повторно.

Так, головним компонентом є AppComponent, що є вхідною точкою клієнтського застосунку. Саме в ньому розміщуються інші компоненти сторінки.

На рисунку 7.15 зображено частину компоненту DosBlockedDevicesPage, що відповідає за відображення сторінки заблокованих пристроїв.

```
const DdosBlockedDevicesPage = (props: DdosBlockedDevicesPageProps) => {
  const [items, setItems] = useState(new Array<SelectableBlockedDevice>());
  const [pageNumber, setPageNumber] = useState(0);
  const [totalPages, setTotalPages] = useState(1);
  const [pageSize, setPageSize] = useState(5);
  const [isLoading, setIsLoading] = useState(false);

  function checkAll(selected: boolean) {
    const newItems = [ ...items];
    newItems.forEach((item) => {
      item.selected = selected;
    });
    setItems(newItems);
  }

  function checkItem(selected: boolean, item: SelectableBlockedDevice) {
    const newItems = [ ...items];
    newItems.find((i) => i.Id == item.Id).selected = selected;
    setItems(newItems);
  }

  function onGridRawClick(i: number) {
    const newItems = [ ...items];
    newItems[i].selected = !newItems[i].selected;
    setItems(newItems);
  }
```

Рисунок 7.15 – Частина компоненту сторінки заблокованих пристроїв

А візуальний результат даного компоненту, тобто інтерфейс користувача для роботи з заблокованими пристроями наведено на рисунку 7.16.

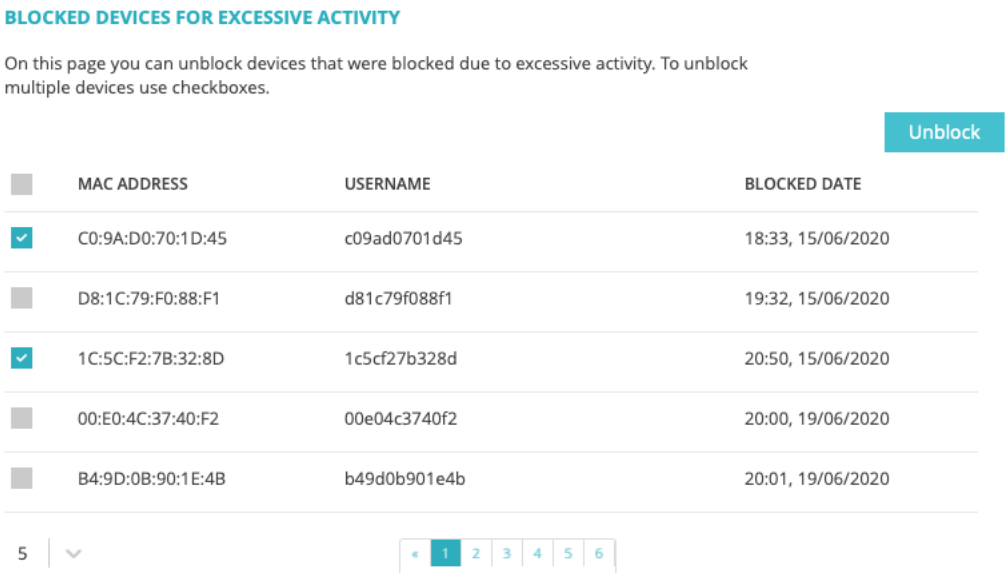

Рисунок 7.16 – Інтерфейс роботи із заблокованими пристроями

Для обміну даними з сервером використовуються запити до API сервера у форматі JSON.

Роботу з даними на стороні клієнта, а саме завантаження заблокованих пристроїв та їх розблокування, зображено на рисунку 7.17

```
import { getJsonDataOrThrow, ajax, ResponsePayload } from "../../../api/ajax";
import * as models from "./types";

export async function loadBlockedDevices(
  pageNumber: number,
  pageSize: number
): Promise<models.GetAllBlockedDevicesResponse> {
  const req = ajax
    .get("/radius/ddos/blocked-devices/list")
    .query({ pageNumber, pageSize })
    .set("Accept", "application/json");

  const response = getJsonDataOrThrow<ResponsePayload<models.GetAllBlockedDevicesResponse>>(
    await req
  );
  return response.Payload;
}

export async function unblockDevices(models: models.BlockedDeviceModel[]): Promise<any> {
  const req = ajax
    .delete("/radius/ddos/blocked-devices/unblock")
    .set("Accept", "application/json")
    .send(models);

  getJsonDataOrThrow<ResponsePayload<any>>(await req);
}

export * from "./types";
```

Рисунок 7.17 – API запити на стороні клієнта

7.5 Висновки розділу

У даному розділі було описано реалізацію спроектованої системи, а саме таких її частин як серверна, клієнтська, мережевий екран та частина RADIUS-сервера.

Для розробки серверної частини було використано мову програмування C#, фреймворк ASP.NET Core, хмарні технології Azure Event Hub, Service Bus та Stream Analytics.

Клієнтська частина було розроблена за допомогою фреймворку React.js та використання компонентного підходу до розробки веб-застосунку.

Також було описано налаштування та розширення конфігурацій серверу автентифікації FreeRADIUS для потреб реалізації розроблюваної системи та реалізацію за допомоги мови програмування C++ модуля надсилання запитів для комунікації FreeRADIUS з технологіями Azure та в подальшому – сервером для захисту.

Мережевий екран було реалізовано за допомогою системи BIG-IP, та таких її можливостей як iRules для фільтрації пакетів, списки груп даних для збереження інформації про заблоковані пристрої та SSL Bridging для роботи з зашифрованим трафіком у випадку RadSec .

8 СТАРТАП ПРОЕКТ

8.1 Опис ідеї проекту

Щоб достовірно визначити можливості даного проекту конкурувати на ринку опишемо ідею розроблюваного програмного продукту у таблиці 8.1.

Таблиця 8.1 – Опис ідеї проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Система, що дозволяє захищати сервер автентифікації RADIUS від можливих DoS атак шляхом фільтрації пакетів та впровадженням порогу на кількість запитів за період часу	Корпоративні мережі	Забезпечує стабільність роботи сервера автентифікації, надає можливість отримувати наглядну інформацію про проблемі пристрої та користувачів та можливість при потребі дозволяти їм доступ, якщо вони були заблоковані попередньо.

Цільова аудиторія проекту складається лише з двох груп:

- бізнеси, що використовують сервер автентифікації RADIUS у своїй корпоративній мережі;
- бізнеси, що реалізують рішення контролю доступу до мережі з використанням сервера RADIUS.

Система буде користуватися попитом тому, що вона є потужним та гнучким інструментом захисту сервера RADIUS у випадку DoS атак. Метою проекту є забезпечення стабільності роботи під час таких атак.

Отже, описавши ідею проекту визначено, що даний проект є перспективним та охоплює велику аудиторію користувачів.

Наступним етапом проведемо аналіз техніко-ідейних характеристик конкурентних продуктів. Потрібно порівняти наявність характеристик серед наявних конкурентів.

Таблиця 8.1 – Визначення характеристик ідеї проекту

Техніко-економічні характеристики ідеї	Конкуренти				W	N	S
	Проект	AirMagnet WiFi Analyzer	Extreme AirDefense	Snort			
Аналіз пакетів	+	+	+	+	+		
Управління пристроями	+	+	–	–			+
Блокування пристрою за потреби	+	+	–	+		+	
Дружній до користувача інтерфейс	+	–	+	–	+		
Підтримка користувацьких налаштувань	+	–	–	+			+

За результатами порівняння характеристик з вищенаведеної таблиці можна зробити висновок, що ідея проекту є конкурентоспроможною.

8.2 Технологічний аудит ідеї проекту

Технологічний аудит проекту дозволяє оцінити доступність реалізації ідеї. Проведемо аналіз технологій, які будуть основними в реалізації ідеї, де будуть ураховуватися такі фактори, як наявність на доступність технологій. Його зображено в таблиці 8.2.

Таблиця 8.2 – Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
Аналіз пакетів	iRules	Наявна	Доступна, але потребує оплати в рамках сервісу Azure
Адміністрування пристроїв	Мова програмування C#	Наявна	Доступна
Робота з даними	API	Наявна	Доступна
Клієнтський портал для адміністрування	Фреймворк React.js	Наявна	Доступна

З зазначеної таблиці можна зробити висновок, що всі вказані технології, які будуть використані в проекті є доступними, але для аналізу пакетів потрібна технологія, що потребує оплати в рамках хмарного сервісу Azure.

8.3 Аналіз ринкових можливостей запуску стартап-проекту

Далі визначимо ринкові можливості стартапу, що можна використовувати під час його ринкового впровадження, а також ринкові загрози проекту, які потенційно можуть зашкодити реалізації проекту.

Це дозволить спланувати майбутні напрями для розвитку проекту з урахуванням стану ринкового середовища, а також потреби потенційних клієнтів проекту та пропозиції конкурентів.

У таблиці 8.3 зазначено потенційних клієнтів, їх характеристики, а також орієнтовний перелік вимог до товару для зазначених груп, які у випадку проекту є системні адміністратори та бізнеси, що розроблюють рішення контролю доступу до мережі.

Таблиця 8.3 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Забезпечення захисту від DoS атак RADIUS-сервера	Системні адміністратори, бізнеси, що розроблюють контроль доступу до мережі	- фінансове положення споживача, можливість його купляти продукт; - технології та пристрої які він використовує;	- легкий у використанні; - швидка робота; - візуальне відображення даних.

Далі проведемо аналіз ринкового середовища.

У таблиці 8.4 зазначено які фактори загроз можуть вплинути на стартап-проект в подальшому.

Таблиця 8.4 – Фактори загроз

Фактор	Зміст загрози	Можлива реакція компанії
Конкуренція	Велика кількість вже існуючих застосунків для моніторингу та захисту корпоративних мереж, що підвищує рівень конкуренції	Створення оригінальних характеристик застосунку у порівнянні з вже існуючими конкурентами.
Швидкий розвиток технологій	Технології розвиваються дуже швидко, а тим самим і створюються нові типи кібер-атак ,яким необхідно буде протидіяти	Оперативне покращення та підлаштовувано під нові потреби

Фактори, що сприяють ринковому впровадженню проекту описано у таблиці 8.5.

Таблиця 8.5 – Фактори можливостей

Фактор	Зміст можливості	Можлива реакція компанії
Широка цільова аудиторія	Аудиторія продукту охоплює практично всі типи бізнесів, що може сприяти широкому розповсюдженню продукту	Створити якісну та потужну рекламу
Інтеграція з іншими сервісами	Так як вже існує багато різних систем для моніторингу та захисту корпоративних мереж, то за допомогою інтеграцій можна збільшити кількість користувачів	Покращення та розширення архітектури для якісного впровадження

Далі проведемо аналіз пропозиції, а саме визначимо загальні риси конкуренції на ринку. Результати аналізу зазначено у таблиці 8.6.

Таблиця 8.6 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1	2	3
1. Вказати тип конкуренції - досконала конкуренція	Компанії застосунків не встановлюють контроль над цінами за використання застосунків, так як кожна	Встановлення ціни на використання застосунку за власним бажанням.

1	2	3
	компанія встановлює власну ціну.	
2. За рівнем конкурентної боротьби - міжнародний	Ринок таких застосунків загалом не є прив'язаним до певної країни, адже бізнеси є усюди, а тим самим і підтримка безпеки корпоративних мереж.	Необхідність правильного таргетингу аудиторії через різні особливості бізнесу.
3. За галузевою ознакою - внутрішньогалузева	Конкуренція не виходить за рамки ринку застосунків забезпечення безпеки корпоративних мереж	Аналіз ринку даної галузі з метою покращення власного продукту.
4. Конкуренція за видами товарів: - товарно-родова	Застосунки на ринку безпеки корпоративних мереж в основному полягають у моніторингу та протидії небажаним діям або пристроям. Вони можуть включати певні спільні риси, але в цілому їх спеціалізація різна.	Поширення функціональності застосунку шляхом включення інших можливостей забезпечення безпеки RADIUS-серверу та корпоративної мережі в цілому
5. За характером конкурентних переваг - нецінова	Застосунки модернізуються з використанням спеціально розроблених методів, поширюється використання різних технологій, що покращують їх використання.	Необхідність креативного та наукового підходу до розробки можливостей застосунку.

1	2	3
6. За інтенсивністю - не марочна	Брендинг не має великої ролі у конкуренції, більше – якість продукції.	Пріоритет на якість розроблюваного продукту.

За результатами аналізу конкуренції проведемо більш детальний аналіз умов конкуренції в галузі за М. Портером. Дана модель зазначає п’ять основних факторів, що випливають на привабливість вибору ринку з оглядом на характер конкуренції. [26]

Сильні позиції компанії за кожним з факторів означають її можливо- сті забезпечити необхідні темпи обороту капіталу та її здатність впливати на інших гравців ринку, диктуючи їм власні умови співпраці.

Дана модель дозволяє дізнатися інтенсивність конкурентної боротьби з боку прямих конкурентів, зробити висновки щодо можливості входу в ринок та його строки, потенційних конкурентів. А також як клієнти та постачальники диктують умови роботи на ринку і які є обмеження для роботи на ринку через товари замінники.

Цей аналіз описано у таблиці 8.7.

Таблиця 8.7 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	-AirMagnet WiFi Analyzer - Extreme AirDefense -Snort	- лояльність клієнтів; - реклама	-	- потреби користувачів; - доступність цін; - якість застосунків.	- лояльність клієнтів; -привабливіша ціна

За наведеною таблицею можна зробити висновок, що інтенсивність конкуренції є доволі високою через вже існуючу популярність та репутацію, але можливості входу на ринок існують.

При розвитку застосунку він зможе стати потенційним конкурентом на ринку та зможе увійти на нього протягом 2-3 років.

А потреби користувачів грають велику роль у розробці продукту, так само як і доступність цін та якість розроблюваних застосунків.

Товари-замінники можуть викликати труднощі на ринку через лояльність вже існуючих клієнтів та кращу цінову пропозицію за використання застосунку.

На основі аналізу складових моделі 5 сил М. Портера розробимо перелік факторів конкурентоспроможності для ринку, що наведено у таблиці 8.8.

Таблиця 8.8 – Обґрунтування факторів конкурентоспроможності

Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
Використання провідних технологій	Використання покращених алгоритмів та скомбінованих методів захисту від DoS атак з подальшим можливим розширенням з штучним інтелектом для більш точного захисту від атак.
Орієнтованість на споживача	Прислухання до споживачів, аналіз їх потреб та впровадження потрібних функцій у застосунок.
Freemium політика	Така політика дозволить отримати основну частину функцій безкоштовно, а за більш розширену версію користувачам треба буде доплатити. Таким чином користувач може поширювати функціональність застосунку за потреби.

Після визначення факторів конкурентоспроможності продукту, проведемо аналіз сильних та слабких сторін стартап-проекту. Його наведено у таблиці 8.9

Таблиця 8.9 – Порівняльний аналіз сильних та слабких сторін проекту

Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з проектом						
		–3	–2	–1	0	1	2	3
Використання провідних технологій	18							+
Орієнтованість на споживача	15						+	
Freemium політика	11				+			

На фінальному етапі ринкового аналізу можливостей впровадження проекту складемо SWOT-аналіз, що є матрицею саналізу сильних та слабких сторін, загроз та можливостей на основі раніше визначених ринкових загроз та можливостей, а також сильних та слабких сторін. Цей аналіз зазначено у таблиці 8.10

Таблиця 8.10 – SWOT- аналіз стартап-проекту

Сильні сторони: - використання провідних технологій; - креативний підхід до розробки функціональності; - клієнт-орієнтованість; - технічна підтримка.	Слабкі сторони: - ціна на підписку; - невелика кількість можливих функцій;
Можливості: покращення алгоритму визначення та захисту від DoS атак; впровадження штучного інтелекту у алгоритм для кращих результатів; - розширення для підтримки захисту інших частин корпоративної мережі.	Загрози: - труднощі впровадження функціональності, що базується на штучному інтелекті; - труднощі в пошуку кваліфікованих спеціалістів для співпраці

8.4 Розроблення ринкової стратегії проекту

Для початку в рамках розроблення ринкової стратегії проекту визначимо стратегії охоплення ринку, а саме опишемо цільові групи потенційних споживачів у таблиці 8.11.

Таблиця 8.11 – Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Бізнеси, які хочуть відслідковувати та попереджати можливі загрози корпоративним мережам	Так як продукт виконує доволі важливу та актуальну роль, то сповивачі готові прийняти даний продукт		Наразі існує велика кількість продуктів, що спрямовані на моніторинг та безпеку корпоративних мереж	Враховуючи доволі високу конкуренцію, вхід у сегмент є не найпростішим, але при правильному маркетингу можна полегшити вхід.

В результаті за цільові групи обрано бізнеси, що хочуть підтримувати якісну роботу корпоративної мережі.

Далі сформуємо базову стратегію розвитку для роботи в обраному сегменті у таблиці 8.12.

Таблиця 8.12 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Стратегія спеціалізації	Концентроване	Акцент на оригінальних можливостях застосунку та клієнт-орієнтованість.	Стратегія диференціації

Обрана стратегія спеціалізації передбачає концентрацію на потребах одного цільового сегменту, без бажання охопити цілий ринок.

Мета полягає в задоволенні потреб вибраного цільового сегменту краще, ніж конкуренти, тобто зробити акцент на оригінальних можливостях застосунку та клієнт-орієнтованість. Проте низька ринкова доля у разі невдалої реалізації цієї стратегії може істотно підірвати конкурентоспроможність компанії.

Також ризиками обраної стратегії є наступні:

– розрив у цінах по відношенню до неспеціалізованого товару конкурентам стає занадто великим;

– різниця у вимогах до товару в межах цільового сегмента та ринку в цілому скорочуються;

– конкуренція виходить на ще більш вузькі підсегменти всередині цілого сегменту.

Далі також зробимо вибір стратегії конкурентної поведінки, що зазначено в таблиці 8.13.

Таблиця 8.13 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Ні, подібні проекти вже існують	В основному – забирати існуючих споживачів у конкурентів.	Так, копіювання буде присутнє, а саме: використання основних методів захисту від DoS атак, фільтрація пакетів тощо.	Стратегія наслідування лідеру

Головна перевага обраної стратегії конкурентної поведінки, це економія фінансових ресурсів, що пов’язані з необхідністю розширення галузевого ринку, постійними інноваціями, витратами на утримання домінуючого положення, адже таке підприємство усвідомлює своє місце і йде у фарватері фірм-лідерів.

Наступним етапом на основі вимог споживачів до стартап-компанії та до продукту, а також в залежності від обраної базової стратегії розвитку, а саме стратегії диференціації та стратегії конкурентної поведінки – стратегії наслідування лідеру – розробимо стратегію позиціонування, що полягає у формуванні ринкової позиції, за яким споживачі будуть ідентифікувати проект.

Результат зазначено у таблиці 8.14.

Таблиця 8.14 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
- зручність у користуванні; - ефективність захисту.	Стратегія диференціації	- Використання провідних технологій - Орієнтованість на споживача - Freemium політика	Якісний, точний, безпечний захист

8.5 Розроблення маркетингової програми стартап-проекту

Спочатку сформуємо маркетингову концепцію товару, який отримає споживач. У таблиці 8.15 для цього було використано результати попередньо проведеного аналізу конкурентоспроможності товару.

Таблиця 8.15 – Визначення ключових переваг концепції потенційного товару

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
Захист сервера автентифікації від DoS атак	Можливість легко та зручно налаштовувати та захищати сервер від атак	- впровадження штучного інтелекту; - комбінування методів захисту від DoS атак.

Далі розробимо трирівневу маркетингову модель товару, тобто зазначимо ідею продукту та його послуги, фізичні складові та особливості його надання. Дана інформація наведена у таблиці 8.16.

Таблиця 8.16 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
I. Товар за задумом	Захист сервера автентифікації від DoS атак за допомогою фільтрації пакетів
II. Товар у реальному виконанні	Властивості/характеристики
	1. Аналіз пакетів 2. Блокування пристроїв з сумнівною активністю 3. Можливе адміністрування з веб-інтерфейсу 4. Використання користувацьких параметрів для блокування
	Якість: швидкість та коректність у використанні, логічність та простота інтерфейсу користувача.
	Пакування відсутнє
III. Товар із підкріпленням	До продажу: відсутнє
	Після продажу: можливість розширити функціональність за допомогою підписки.
За рахунок чого потенційний товар буде захищено від копіювання: створення торгової марки та впровадження авторського права.	

Наступним кроком визначимо оптимальну систему збуту у таблиці 8.17.

Таблиця 8.17 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Збут через Інтернет-платформи	- розміщення застосунку на власному маркеті; - реклама застосунку на головних сторінках;	1	Залучена система збуту через сторонніх посередників

За результатами таблиці оптимальною системою збуту є збут через сторонніх посередників з глибиною каналу збуту 1.

Останнім етапом розробимо концепцію маркетингових комунікацій, що буде опиратися на попередньо обрану основу для позиціонування та визначену специфіку поведінки клієнтів у таблиці 8.18.

Таблиця 8.18 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Можуть попросити додати певні нові функції для подальшого користування	Соціальні мережі, конференції	За сферою застосування; показники якості; позитивні особливості технології	Заохотити до підтримки безпеки корпоративної мережі	SMM

8.6. Висновки розділу

У даному розділі було розглянуто ідею стартап проекту на основі розроблюваного програмного забезпечення.

Головною цільовою групою є бізнеси, які хочуть відслідковувати та попереджати можливі загрози корпоративним мережам.

Аналіз технічної здійсненності проекту показав, що всі вказані технології, які будуть використані в проекті є доступними, але для аналізу пакетів потрібна технологія, що потребує оплати в рамках хмарного сервісу Azure.

Також було проведено аналіз ринкових можливостей проекту та створені маркетингова та ринкова стратегії просування цієї системи. Було обрано стратегії диференціації та наслідування лідеру.

Серед факторів конкурентоспроможності було обрано використання провідних технологій, орієнтованість на споживача та Freemium політика

Було визначено такі загрози, як конкуренція та швидкість розвитку технологій, а також можливості – широка цільова аудиторія та можливі інтеграції з іншими сервісами.

Проект має перспективи на подальший розвиток та має багато переваг серед аналогів, проте конкуренція також присутня, тому для успішного виходу на ринок необхідно мати гарну якість застосунку та швидко реагувати на необхідні покращення.

ВИСНОВКИ

У ході виконання роботи була розроблена та створена система захисту сервера автентифікації RADIUS від DoS атак.

Дана система допомагає захисти сервер RADIUS від можливих DoS атак шляхом фільтрації пакетів.

У ході виконання роботи було проведено дослідження предметної області та проаналізовано існуючі аналоги системи, що була розроблена, та методи визначення та захисту від DoS атак.

На основі характеристик цих аналогічних рішень, порівняння їх переваг та недоліків було сформовано функціональні та не функціональні вимоги до розроблюваної системи. Використовуючи дані вимоги було створено діаграму сценаріїв використання та детально описано кожен із зазначених сценаріїв.

В роботі було описано алгоритм захисту від DoS атак, що в основному являє собою фільтрацію пакетів, що використовує дані з запиту, а саме ім’я користувача, MAC-адресу пристрою, порт на який цей запит прийшов та порогове значення для кількості запитів за певний період часу. Таким чином розроблений алгоритм включив у собі деякі характеристики описаних методів визначення та захисту від DoS атак, а саме метод, що базується на порозі, метод виділення певних ознак, фільтрація MAC-адрес та фільтрація пакетів. Тим не менш, його можна ще покращити шляхом впровадження нейронної мережі та штучного інтелекту.

Для розробки було проаналізовано технології, які можна використовувати для створення системи.

В результаті аналізу було обрано наступні: мову програмування C#, а саме фреймворк .NET Core для реалізації серверної частини та бізнес логіки, та React.js для реалізації клієнтського веб-інтерфейсу.

Для збереження даних було обрану реляційну базу даних MySQL.

Для серверу автентифікації було обрано вільний для використання сервер FreeRADIUS, а для реалізації допоміжних функцій спілкування серверу захисту та серверу автентифікації було використано мову програмування C++.

Мережевий екран було реалізовано за допомогою системи BIG-IP, а особливо її можливостей iRules для фільтрації пакетів та групи даних для збереження інформації про заблоковані пристрої. Також в рамках мережевого екрану для роботи з зашифрованим трафіком було реалізовано такий підхід як SSL Bridging, що дозволяє розшифровувати трафік для обробки та зашифровувати його для подальшого надсилання.

Для надсилання даних до серверу системи захисту, отриманих з серверу автентифікації, та початкової стадії аналізу – агрегації потрібних даних – було використано технології хмарної платформи Microsoft Azure, а саме Event Hub, Service Bus та Stream Analytics.

Також було описано як дана система може бути реалізована в якості стартапу в останньому розділі. Була розроблена маркетингова та ринкова стратегії просування даного проекту.

За результатами цього розділу було зроблено висновок, що проект має перспективу для виходу на ринок та подальший розвиток, хоча конкуренція є присутньою. Але головні фактори, що дозволять подальший розвиток є гарна якість застосунку та швидкість в реагуванні на необхідні покращення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What Is Network Access Control? Solutions and Explanations[Електронний ресурс]. — Режим доступу: https://www.varonis.com/blog/network-access-control-nac/

2. Remote Authentication Dial In User Service (RADIUS) [Електронний ресурс]. — Режим доступу: https://tools.ietf.org/html/rfc2865

3. RADIUS Accounting [Електронний ресурс]. — Режим доступу: https://tools.ietf.org/html/rfc2866

4. TLS encryption for RADIUS over TCP (RadSec) [Електронний ресурс]. — Режим доступу: https://tools.ietf.org/id/draft-ietf-radext-radsec-00.html

5. M.Voznak. DoS Attacks Targeting SIP Server and Improvements of Robustness / M.Voznak., J. Safarik / International Journal Of Mathematics And Computers In Simulation. – 2012. – vol. 6, is. 1. – P. 177-184

6. Анализатор Fluke Networks AirMagnet WiFi Analyzer AM/A1150 [Електронний ресурс]. — Режим доступу: https://www.protehnology.ru/analizator-fluke-networks-airmagnet-wifi-analyzer

7. Fluke Networks AirMagnet WiFi Analyzer Pro – программное обеспечение для Wifi [Електронний ресурс]. — Режим доступу: https://iron-harry.ua/tovar/50597/

8. Extreme AirDefense [Електронний ресурс]. — Режим доступу: https://www.extremenetworks.com/product/extreme-airdefense/

9. SNORT Users Manual [Електронний ресурс]. — Режим доступу: http://manual-snort-org.s3-website-us-east-1.amazonaws.com

10. Формирование требований и классификация требований [Електронний ресурс]. — Режим доступу: https://analytics.infozone.pro/formation-requirements-and-classification-requirements/

11. Начало работы с .NET Framework [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/

12. FreeRADIUS [Електронний ресурс]. — Режим доступу: https://ru.wikipedia.org/wiki/FreeRADIUS

13. Хмарні рішення Microsoft Azure [Електронний ресурс]. — Режим доступу: http://integritysys.com.ua/solutions/privatecloud-solution-azure/

14. React или Angular или Vue.js — что выбрать? [Електронний ресурс]. — Режим доступу: https://habr.com/ru/post/476312/

15. SSL Bridging [Електронний ресурс]. — Режим доступу: https://www.f5.com/services/resources/glossary/ssl-bridging

16. Relational VS Non-Relational Databases [Електронний ресурс]. — Режим доступу: https://medium.com/@zhenwu93/relational-vs-non-relational-databases-8336870da8bc

17. ACID [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/ACID

18. Relational vs. non-relational databases: Which one is right for you? [Електронний ресурс]. — Режим доступу: https://www.pluralsight.com/blog/software-development/relational-non-relational- databases

19. Representational state transfer [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Representational_state_transfer

20. Azure Event Hubs — A big data streaming platform and event ingestion service [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-about

21. What is Azure Service Bus? [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-messaging-overview

22. Welcome to Azure Stream Analytics[Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction

23. BIG-IP Application Delivery Services [Електронний ресурс]. — Режим доступу: https://www.f5.com/products/big-ip-services

24. iRules Home [Електронний ресурс]. — Режим доступу: https://clouddocs.f5.com/api/irules/

25. Intermediate iRules: Data-Groups [Електронний ресурс]. — Режим доступу: https://devcentral.f5.com/s/articles/intermediate-irules-data-groups-20430

26. Розроблення стартап-проекту [Електронний ресурс] : Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016. – 28 с.

ДОДАТОК А

```
when CLIENT_ACCEPTED {
   set client_ip [IP::client_addr]
   set client_port [UDP::local_port]
   set username [RADIUS::avp 1 "string"]
   set calling_station_id [RADIUS::avp 31 "string"]
   set key "$username:$client_port:$calling_station_id"
   if {  [class match $key eq device_block_list]} {
reject
      log local0. "rejected $key"
   }
}
```

Лістинг А.1 – Функція випадку обробки UDP трафіку

```
when CLIENTSSL_HANDSHAKE {
   SSL::collect
}
when CLIENTSSL_DATA {
   set payload [SSL::payload]
   set client_port [TCP::local_port]
   set username ""
   set calling_station_id ""
   binary scan $payload cH2SH32cc code ident len auth attr_code1 attr_len1
   # Access-Request or Accounting-Request
   if { $code == 1 || $code == 4 } {
      set index 22

      while { $index < $len } {
         # User-Name attribute
         if { $attr_code1 == 1 } {
            binary scan $payload @${index}a[expr { $attr_len1 - 2 } ]cc attr_value attr_code2
attr_len2
            set username $attr_value
         # Calling-Station-Id attribute
         } elseif  { $attr_code1 == 31 } {
            binary scan $payload @${index}a[expr { $attr_len1 - 2 } * 2 + 2 ]cc attr_value
attr_code2 attr_len2
            set calling_station_id $attr_value
         }
         set index [ expr { $index + $attr_len1 } ]
         set attr_len1 $attr_len2
         set attr_code1 $attr_code2
      }
      # Process mac address
      if { [string length $calling_station_id] == 0 } {
```

```
        set calling_station_id "00:00:00:00:00:00"
      }
      else {
        set calling_station_id [string map {"-" ":"} [string tolower $calling_station_id]]
      }
      set key "client_port:$calling_station_id:$username"
      if { [class match $key eq device_block_list] } {
        SSL::release
        reject
        log local0. "rejected $key"
        return
      }
    }
    SSL::release
    SSL::collect
}
```

Лістинг А.2 – Функція випадку обробки RadSec трафіку

```
 public void BlockDevices(IEnumerable<int> ports, IEnumerable<BlockedDeviceInfo>
devices)
      {
         var deviceKeys = GetBlockedDevicesKeys(ports, devices);
         var cmd = new BigIpCmdBuilder()
                .AddToBlockedDataGroupList(deviceKeys)
                .AddSave()
                .Build();
      }
```

Лістинг А.3 – Приклад методу додавання ключа в групу даних

```
function main() {
   this.init = function () {
      this.enableDiagnostics = true;
      this.result = {
         Kind: 0,
         TimestampUnix: null,
         TotalCounters: {},
         CountersByType: {
         },
         Diagnostics: {
            AggregatedAlerts: [],
            Errors: []
         },
         TotalUpdates: 0
      };
   };
```

```
    // windowEnd argument is required, otherwise UDA would be considered as invalid
    this.accumulate = function (alert, windowEnd) {
        this.execute(() => this.collectDiagnostics(alert));
        let diff = 0;
        if (alert.AlertType === this.AlertTypes.RadiusSuccessAuthRequest
            || alert.AlertType === this.AlertTypes.RadiusFailedAuthRequest
            || alert.AlertType === this.AlertTypes.RadiusAcctRequest) {
            diff = 1;
        }
        const { UserName, DeviceMac } = alert.Details || {};
        if (!UserName || !DeviceMac) {
            diff = 0;
        }
        if (diff !== 0) {
            const key = `${UserName}_${DeviceMac}`;
            this.execute(() => this.updateCounter(this.result.CountersByType, key, alert.AlertType));
        }
    };
    this.updateCounter = function (counters, name, type) {
        if (!counters[name]) {
            counters[name] = {};
        }
        counters[name][type] = (counters[name][type] || 0) + 1;
        this.result.TotalUpdates++;
    };
    this.computeResult = function () {
        const hasErrors = this.result.Diagnostics.Errors.length > 0;
        if (!hasErrors && this.result.TotalUpdates === 0) {
            return null;
        }
        this.compressDiagnostics(hasErrors);
        return this.result;
    };

    this.collectDiagnostics = function (alert) {
    if (!this.enableDiagnostics) {
        return;
    }
    this.result.Diagnostics.AggregatedAlerts.push(data);
};
    this.compressDiagnostics = function (force) {
    if (force || this.enableDiagnostics) {
        const aggregated = this.result.Diagnostics.AggregatedAlerts;
        // exclude aggregated alerts
```

```
        // if resulting Service Bus message size is going to exceed 256 KB
        if (aggregated.length > 8 && JSON.stringify(aggregated).length > 200000) {
            this.result.Diagnostics.AggregatedAlerts = [];
        }
    }
};
    this.execute = function (action) {
    try {
        action();
    }
    catch (ex) {
        // swallow exception: if some part of UDF fails
        // other calculations would not be affected
        if (this.enableDiagnostics) {
            this.result.Diagnostics.Errors.push(String(ex));
        }
    }
};
    this.ticksToUnixTime = function (ticks) {
    // subtract number of ticks in Unix epoch (1970-01-01) and convert to seconds
    // there are 10 million ticks in one second
    return Math.floor((ticks - 621355968000000000) / 10000000);
};

    this.AlertTypes = {
        RadiusSuccessAuthRequest: 1013,
        RadiusFailedAuthRequest: 1014,
        RadiusAcctRequest: 1015
    };
}
```

Лістинг А.4 – Приклад функції Azure Stream Analytics

```
SELECT uda.countRadReqPerDevice(radRequestsAlertsPerDeviceCounter)
INTO [counter-sb-output-radrequests-per-device]
FROM radRequestsAlertsPerDeviceCounter
GROUP BY TUMBLINGWINDOW(minute, 2)
```

Лістинг А.5 – Приклад SQL-запиту для агрегації даних

```
  public async Task HandleAsync(BaseTimeCounters counter)
        {
            try
```

```
        {
            var blockingSettings = await _deviceBlockingSettingsDA.GetAsync();
            // only without enabled DisableBlocking option
            if (blockingSettings.DisableBlocking)
            {
                return;
            }
            // check counter values
            var devicesToBlock = GetDevicesToBlock(counter, blockingSettings);
            if (devicesToBlock.NotEmpty())
            {
            devicesToBlock = await
_devicesBlocker.BlockDevicesAsync(devicesToBlock);
            }

            if (devicesToBlock.NotEmpty())
            {
            await NotifyOfBlockedDevices(devicesToBlock, blockingSettings);
            }
        }
        catch (Exception ex)
        {
            Log.TraceError(Errors.AsaCounter_DosPerDeviceProcessFailed, ex,
logProperties, new[] { counter.CountersByType?.ToJsonNet() });
        }
    }
```

Лістинг А.5 – Метод фонового процесу обробки даних на сервері